G. FRIEDLAND, R. JAIN

FUNDAMENTALS OF MULTIMEDIA COMPUTING

DRAFT FROM NOVEMBER 13TH, 2011

LAST EDITS: G.F.

GLOBAL TODOS:

- Agree on reference styles and also on reference categories (literature, web links, or what?)

- Rethink labeling of images and tables

- Replace all XXX with actual labels.

Dedication and other stuff

Chapter 1: Introduction

When people think of multimedia computing, they usually think of video in a computing environment. This is a narrow perspective on multimedia. Visual information definitely dominates human activities because of the powerful visual machinery that we are equipped with. But, humans use all five senses effectively, opportunistically, and judiciously. A true multimedia system should be able to effectively utilize signals from multifarious sensors and present to users only the relevant data in the appropriate media.

This book takes an *integrative systems* approach to multimedia. Integrated multimedia systems receive input from different sensory and symbolic sources in different forms and representations. Users ideally access this information in experiential environments. Early techniques dealt with individual medium more effectively than with integrated media and focused on developing effective techniques for separate individual medium, e.g. MPEG video compression. As the field matures, we are seeing increasing attention on issues that span multimedia. As we will discuss in more detail later, most of the difficult semantic issues become easier to solve when considering integrated multimedia rather than separate individual medium.

In the early days of computing, science fiction writers envisioned computers as robots that would effectively use audio-visual data. Later, people dreamt of systems that could organize audio files, images, and video. Now people want to share perceptual experiences independent of time and distance. Within the next few years, most of the data stored on computers—at least with regard to storage size and bandwidth requirements—will be audio-visual. The fundamental medium for computing and communications will also be audio-visual, and most likely also tactile.

Handling multimedia content requires incorporating concepts and techniques from various disciplines—from signal processing, from communication theory to image databases, and from compression techniques to content analysis. Multimedia computing has consequently evolved as a collection of techniques from different disciplines.

Unfortunately, both application development as well as the multimedia research field has evolved in a way like the elephant in the fable about the elephant and the six blind men (see Figure 1). In this fable, each blind man has a limited perspective due to some physical limitation. In

real life, people impose limitations of perspective in many ways and hence—though naturally endowed with multiple sensory and cognitive faculties—functionally behave like these blind men portrayed in the cartoon: Each engineering and research discipline perceives multimedia from its own limited viewpoint. This has resulted in a skewed development of the field, where multimedia is perceived — at best — as multiple monomedia fields.

We (humans) use our five senses (sight, hearing, touch, smell, and taste) together with our abstract knowledge to form holistic experiences and extract information. Multimedia computing aims to develop communication techniques to allow holistic experiences from multiple sources and modalities of data and extract useful information in the context of various applications.

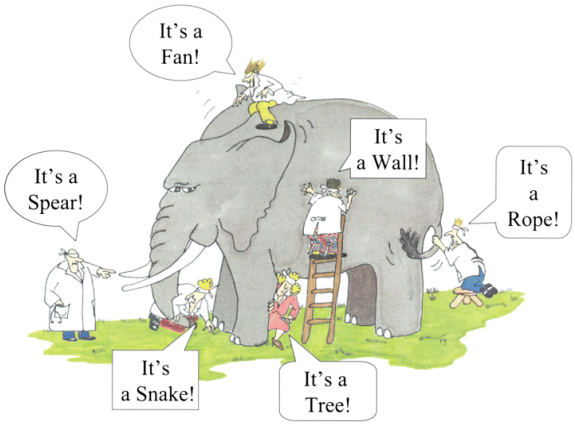


Figure 1: Multimedia is like an elephant. Looking at it from limited perspective leads to many completely wrong characterizations.

This fragmented perspective of multimedia has slowed progress in understanding and effectively processing multimedia information, although the hardware used for processing it—ranging from sensors to bandwidth—has advanced rapidly. Multimedia computing should leverage correlated and contextual information from all sources to develop holistic and unified perspectives and experiences. It should focus on full multisensory experiences rather than partial experience, such as listening to a an audio-only sports commentary.

This book presents emerging techniques in multimedia computing from an experiential perspective in which each medium—audio, images, text, and so on—is a strong component of the complete exchange of information or experience. Humans are the best functioning example of multimedia communication and computing—that is, we understand information and experiences

through the unified perspective offered by our five senses. Our goal in this book is to present current techniques in computing and communication that will lead to the development of a unified and holistic approach to computing using heterogeneous data sources.

By describing the properties of perceptually encoded information, presenting common algorithms for handling it, and outlining the typical requirements for emerging applications that use multifarious information sources, this book introduces the fundamentals of multimedia computing. It serves as an introduction to engineers and researchers interested in understanding the elements of multimedia and their role in building specific applications.

Organization of this Book

We organized this book to present a unified perspective on different media sources for addressing emerging applications. The book consists of six parts, each containing several chapters. We provide pointers to the latest literature, but our main goal is to present concepts, techniques, and applications that will be useful in building integrated multimedia systems. We believe that the holistic viewpoint presented in this book is essential for understanding, using, and communicating emerging applications that use heterogeneous data from multifarious sources.

1. Defining Multimedia Systems

The current stage of the multimedia field brings to mind the parable about the six blind men and the elephant; we therefore define multimedia systems and discuss its main elements. This will help us discuss all elements concurrently without losing the whole-system perspective.

2. Nature of Perceptually Encoded Information

Like humans, multimedia systems gain information and experience through a variety of sensory and other sources. Understanding the relationships among data, information, knowledge, insight, and experience is crucial to being able to use these sources judiciously. We discuss basic elements of information and data source types, including text, audio, images, and video, in the context of multimedia systems. These areas are well established and many other sources provide details on every aspect of representation and processing. Our goal here will be to present the essential elements from those areas and direct readers to sources for more information.

3. Fundamental Properties of Multimedia Applications and Systems

Once multimedia data is acquired through sensors, it needs to be transmitted, stored, reproduced. Users often use production environments to edit and create multimedia presentations out of the raw sensor data. This part of the book discusses the fundamentals concepts of multimedia systems and applications.

4. Compression

Sensors are often located at geographical locations outside of the processing environment. Users, too, are typically at different geographic locations from the processor. Thus, increasingly, a system's input, processing, and output elements are at different locations. A large volume of data must therefore be communicated to different locations, making data-compression techniques essential. Fortunately, data compression is an active research area and most of these techniques have responded well to multimedia systems' needs. This book presents fundamental algorithms and ideas that allow the reader to go into further details based on his or her interest.

5. Organization and Analysis of Multimedia Content

Multimedia systems require a large amount of storage. Data's distributed nature and presentation issues make storage techniques and architectures serious concerns.

Organizing multimedia data for search and navigation has been a challenge. Even organizing individual components such as audio, images, and video presents challenges. In the last few years, researchers have begun focusing on spatiotemporal multimedia data. Systems that handle this data will require different access environments and different navigation and presentation mechanisms from those used in current databases and search engines.

Index Terms

Elephant parabel; organization, book

Literature

- John Godfrey Saxe: The Blind Men and the Elephant (retold)

PART I:
DEFINING
MULTIMEDIA COMPUTING

Chapter 2: Multimedia: A Definition

Only a few inventions in the history of civilization have had the same impact on society in so many ways and at so many levels as computers. Where once we used computers for computing with simple alphanumeric data, we now use them primarily to exchange information and communicate. Computers are rapidly evolving as a means for gaining insights and sharing experiences across distance and time.

Multimedia computing started gaining serious attention from researchers and practitioners in the 1990s. Before 1991, people talked about multimedia, but the computing power, storage, bandwidth, and processing algorithms were not ready to deal with audio and video. With the increasing availability and popularity of CDs, people became excited about creating documents that could include not only text, but also images, audio, and even video**.** That decade saw explosive growth in all aspects of hardware and software technology related to multimedia computing and communication. In the early 1990s, PC manufacturers labeled their high-end units containing some advanced graphics *multimedia* computers. That trend disappeared a few years later because every new computer was a multimedia computer.

Research and development in multimedia-related areas has been around for much longer. Research in speech processing, speech compression, and speech recognition was fueled first by telephony and then by digital sound applications. Image and video processing and compression have also been active research and development areas due to digital photos and then digital video.

Before 1990, much of the research in audio and video compression, storage, and communication was driven by broadcast and consumer electronics related to entertainment applications. In the 1990s, the idea of combining these sources in a computing environment emerged as a clear possibility. As a result, research in all areas of audio and video received significantly greater emphasis.

In the following we describe the historic evolution of multimedia computing based on the fundamentals of technological advancements and the demands by its users. Knowledge about

these fundamentals contributes significantly to the understanding of the field and enables visions about it's future.

Communication in Human Society

The ability to effectively communicate complex facts and interrelationships is one of the main features that distinguishes humans from animals and has been a major force in human evolution. Communication lets us share experiences and create, maintain, sustain, and propagate knowledge. As Table 1 shows, human civilization has seen many influential inventions related to communicating experiences across space and time. Communication invariant to space allows to exchange information between participants independent of their current location, invariance to time allows to experience an event over and over again without having to be there at the exact moment. Mankind's striving for both time and space invariance in communication is one of the defining foundations of multimedia.

Invention	Resulting Application	Invariance
Spoken Languages	Natural communication	none
Written Languages	Symbolic record of language	Time
Paper	Portability	Time and space
Print	Mass distribution	Time and space
Telegraph	Remote narrow communication	Space
Telephone	Remote analog communication	Space
Radio	Broadcasting of sound	Space
Film	Recording of sight and sound	Time and space
Television	Broadcasting of sight and sound	Space
Recording media	Recording	Time and space
Digital media	Machine enhancement and processing	Time and space
Internet	Personalized reception	Time and space

Table 1: Communications-related inventions in human civilization.

Human communication exists in many forms, including facial and body gestures, olfactory signs, and of course spoken language. Out of these, spoken language is the one that is able to convey the most complex facts, i.e. information density is very high. For most of human's existence though spoken language only consisted of analog sounds uttered with the speech-producing

infrastructure in the the throat. Eventually, people realized that experiences were important and should be stored for sharing with others. Initially, drawings and paintings would convey these experiences but they were not precise enough to inambigously convey complex facts — and too cumbersome to produce. Humankind thus invented written language as a system for representing language so other people could also share experiences. Cumbersome techniques such as stone tablets gave way to more practical storage devices and writing methods. Next came the development of paper and ink, and still more people began using the stored experiences that others had painstakingly recorded.

Then came one of the most influential inventions in our history: Gutenberg's movable printing press. This invention enabled mass communication for the first time, and revolutionized society. Our current education system, our reliance on documents (such as newspapers) as a major source of communication, and libraries as public, government-supported institutions dedicated to storing knowledge, all stem from that one invention that appeared more than 500 years ago.

The telegraph, which allowed instantaneous communication of symbolic information over long distances, began to bring the world closer. This invention signaled the beginning of the global village. Telephones let us return to our natural communication medium—talking—while retaining the advantages of instantaneous remote communication. People could experience the emotions of the person on the other end of the connection—something symbol-based methods of writing and telegraph could only hint at.

Radio ushered in the wireless approach to sound and popularized sound as a medium for instantaneous mass communication. Film and Television took communication a step further by appealing to our sense of sight as well as hearing. It was the first medium that let us experience with more than one sense and as such was able to more effectively key into our emotions. Video communication's popularity is clearly due to its use of our two most powerful senses working in harmony to communicate experiences.

Storage and distribution technologies, such as magnetic tape, allowed the storage, preservation, and distribution of sound, again bringing us closer to natural experience. Video recording enhanced this experience significantly. Digital media further improved the quality of our experience by making it possible to copy and share information without loss. The Internet took infor-

mation availability to a new dimension, providing us with experiential accounts of an unprecedented variety.

Evolution of Computing and Communication Technology

The changes in the landscapes of both computing and communications have been overwhelming in the last few decades.

Just a few decades ago, a computing center was one of the most important buildings on a university or corporate campus. Access to this building, particularly to the "holy" room in which the computer functioned, was highly restricted. A computer occupied several rooms, if not floors, of a building, needed air conditioning, and required a specialized and trained staff to interact with it. These computers cost millions of dollars. Figure 2 shows a popular computer from the late 1960s and early 1970s. Table 2 lists some of its important characteristics.

Figure 2: A 1960s-era computer.

Computer	Processing unit	Operating system	Core memory	Secondary memory
1960s-era computer	Could not do arithmetic, used look-up tables	No OS; human monitors controlled everything	60 Kbytes	2M characters

Modern handheld computer (iPhone)	ARM 620 MHz	iPhone OS	128 Mbytes	16 Gbytes

Table 2: Comparison of early computers with those of a typical handheld computer.

Progress in processing, storage, networking, and software technology have changed computing beyond anyone's expectations. Today, most people carry computers that are more powerful and sophisticated than the 1960s-era computer in their pockets. Figure 3 shows one such computer; Table 2 compares it to the early computer. Although this computer is more powerful and sophisticated than the one in Figure 2, it costs several thousandths of what the older version cost, is easy to carry, and is much less affected by climate. Moreover, just about anyone can operate it, using it to solve their everyday computing and communications needs.

Figure 3: A handheld computer, similar to those most people carry.

Communications technology has experienced a similar overwhelming transformation. We've already discussed the historical perspective. Here, we focus on short-term technological improvements in one medium.

Consider the telephone. In its very early incarnations, the telephone had limited use. Only a few people could afford to have one in their homes. Moreover, a house had one phone, and when you called someone you literally had to shout into the mouthpiece. During a long-distance call, latency made communication difficult. Either both parties spoke at the same time or each waited for the other, while an expensive meter ticked off seconds. Not seldomly, people spent more time shouting "Hello! Hello!" than having a meaningful conversation. Now, users can talk on a phone while walking, running, driving, or flying in an airplane. Signal reception is so clear that you can

whisper to a person on the other side of the globe. More importantly, not only is your phone a voice communication device, but it is also your connection to a computer network, a camera, your calendar and address book, an entertainment, and a video communication device.

The Future of Multimedia: Experiential Computing

To understand computing technology's evolution to its current state, as well as to project its future evolution, consider the applications that have been and will be driving the technology's development.

The first computer applications performed numerical computations using data in scientific applications, hence the name *computer*. Business was the next major driving application with so-called "data processing." It brought alphanumeric processing and databases in focus for development. Major networking advances resulted in enterprise computing based on the traditional distributed processing approaches that eventually culminated in the Internet.

Personal computers were another major influence on computing. PCs ended reliance on a powerful central computer and put several applications, including word processing, spreadsheets, and electronic mail, into the public arena. Combining personal computing and Internet connectivity led to one of the most amazing communication revolutions that human civilization has seen so far: the World Wide Web. The progress continued and laptop computers replaced most PCs. Laptops are now being replaced by an even more personal and sentient computing device— tablets and mobile phones. These mobile devices can be used for computing, communication, and much more. Moreover, they can use audio and visual mechanisms equally effectively as traditional alphanumeric computing. More and more they are being equipped with more diverse sensing mechanisms than humans have, e.g. GPS receivers to sense geo-location. These devices are true multimedia computing and communication devices.

Emerging computing and communication applications have clear differences from earlier applications. This allows us to speculate about the future of multimedia applications and provide a framework for new ideas. The following is a list of properties we observed looking at recently emerging applications:

- Spatiotemporal and live data streams are the norm rather than the exception.
- A holistic picture of an event is more important than silos of isolated data.

- Users want insights and information that are independent of the medium and data source. That is, the medium is just the medium; the message is what's important.
- Users do not want information that is not immediately relevant to their particular interests.
- Exploration (browsing), not querying, is the predominant mode of interaction.

Currently emerging applications are pushing computing to use primarily data from multiple sources. Moreover, these applications clearly demand that computing focus more on information, experiences, and understanding than on the medium or data source.

This evolving nature of data sources and desired operations can be captured in the matrix shown in Table 3. These relationships have profound implications for information and communication technologies (ICT). For example, databases are excellent for getting precise information from a single alphanumeric data destination. Visualization environments and interactive tools combined with data warehousing technology are useful in gaining insights from a precise alphanumeric source. In the last few years, search engines have made tremendous progress in finding information sources, particularly alphanumeric sources in the World Wide Web, which is primarily an unstructured distributed information source. Going forward, most emerging applications will fall in the top-right quadrant: To gain insights from multiple heterogeneous sources, we need an experiential environment because it unites disparate data sources and frees decision-makers to explore their perceptions.

Insight	Visualization	Experiental Environments
Information	Databases	Search Engines
	Single Data Destination	**Multiple Data Destination**

Table 3: The changing nature of applications.

Current information environments often actually work against the human-machine synergy. Humans are very efficient in conceptual and perceptual analysis but relatively weak in mathe-

matical and logical analysis; computers are exactly the opposite. In an experiential environment, users *directly use their senses to observe data and information of interest related to an event, and they interact naturally with the data based on their particular set of interests in the context of that event.*

Experiential environments several important characteristics:

They are direct. An experiential environment provides a holistic picture of an event without using unfamiliar metaphors and commands. People are in a familiar environment and use natural actions based on commonly used operations and their anticipated results. In experiential environments, users easily and rapidly interpret the data presented and then interact with the dataset to get a modified dataset.

They provide the same query and presentation spaces. Most current information systems use different query and presentation spaces. Popular search engines, for example, provide a box for entering keywords, and the system responds with a list of perhaps thousands of entries spanning hundreds of pages. A user has no idea how the entries on the first page relate to those on the 13th, how many times the same entry appears, or even how th*e entries on the same page relate to each other*. Contrast this to a spreadsheet. A user articulates a query by changing certain data that is displayed in the context of other data items. This action results in a new sheet showing new relationships. Here, query and presentation spaces are the same. These systems are called What-You-See-Is-What-You-Get (WYSIWYG).

They consider both the user state and context. Systems should know the user's state and context and present information that is relevant to the user in that state and context. People operate best when they are in known contexts and they do not like instantaneous context switching. Information systems, including databases, should be scalable and efficient. These considerations led to the design of *stateless* systems, such as relational databases. However, this statelessness is why most Internet search engines are so dissatisfying. They don't remember the user's state. *They promote perceptual analysis and exploration.* Text-based systems provide abstract information in visual form. Experiential systems let user analyze, explore, and interact with their environment using all of their senses, and thus are more compelling and easier to understand.

Multimedia systems will play a key role in creating experiential environments in diverse applications. Currently, video games provide the best experiential environments. Often, these games effectively use audio, video, and tactile media to create compelling interactive environments.

Multimedia: A More Formal Definition

One obvious question that comes to mind at this stage is: Are there some fundamental issues in multimedia computing and communication systems that will provide this integrative perspective? For exploring this, let us consider the problem a bit more precisely.

Consider a system equipped with multiple sensors working in a physical environment. The system continuously receives information about the environment from multiple sensors and uses this information to achieve its goals.

Assume that $S_1, \ldots S_n$ are synchronized data streams from sensors. These data streams have K types of data in the form of image sequence, audio stream, motion detector, and so on. Further, let $M_1, \ldots, M_n$ be metadata, including annotations, for each stream. This metadata might include the sensor's location and type, viewpoint, angles, camera calibration parameters, or any other similar parameters relevant to the data stream. In most cases, feature detectors must be applied to each data stream to obtain features that are relevant in a given application. Let us represent feature stream F_{ij}, where F_{ij} is the jth feature stream from S_i.

Multimedia computing and communication techniques combine the dataset S_i and its feature stream F_{ij} using the metadata M_i to extract information about the environment required to solve a given problem. In this process, the system must often combine partial, sometimes uncertain, information from multiple sources to get more complete and reliable information about the environment.

A defining difference in multimedia from single medium understanding fields like computer vision or audio processing is that partial information from multiple media is correlated and combined to get complete information about the environment. A common experience that most people have is deciding about a thunder and explosion—appearance of a bright light followed by a strong sound is used to detect it. Without correlating the sound with the noise, one cannot conclude that there is an explosion or a thunder.

As we will see, context added by both the senses and prior experience plays a key role in human multimedia analysis. In multimedia computing the context can come from anywhere, e.g. from some data collection parameters, from other sensory data, or from device constraints.

Index Terms

Multimedia, Definition of; Multimedia, Evolution of; Experiental Computing; Communication , Fundamentals of

Literature

Research Articles

Web Links

Exercises

1. Think about further techniques developed in history that influenced communication and enabled time and space invariance.
2. Choose a multimedia application of your choice and brainstorm in a group or with a partner how it could be made more experiential.
3. Search several definitions of the word “Multimedia” on the web or in other literature and discuss their meaning based on different contexts.

Chapter 3: Elements of Multimedia Computing

As discussed in the previous chapter, multimedia is closely related to how humans experience the world. In this chapter first we introduce the role of different sensory signals in human perception for understanding environments to function in it and for communicating and sharing experiences. A very important lesson for multimedia technologists is that each sense provides only partial information about the world. Data and information from different sensors must be combined with other senses and prior knowledge to understand the world. One sense alone, even the very powerful sense of vision, is not enough to understand the world. In multimedia computing also, different sensory media should be combined with other knowledge sources to interpret the situation. *Multimedia computing and communication is fundamentally about combining information from multiple sources in the context of the problem being solved*. This is what distinguishes multimedia from several other disciplines, including computer vision and audio processing, where focus is on analyzing one medium to extract as much information as could be extracted from it.

In multimedia systems, different types of data stream simultaneously exist and the system must process them not as separate streams but as one correlated set of streams that represent information and knowledge of interest for solving a problem. The challenge for a multimedia system is to discover correlations that exist in this set of multimedia data and combine partial information from disparate sources to build the holistic information in a given context.

Experience and Information

We experience our physical environment through our natural senses of sight, hearing, touch, taste, and smell[1]. Every human child starts building models of different objects and events in the world through learning process from very early part of life using all senses. Once these models are in place, our senses let us experience and function in the physical and social worlds and refine, enhance and even develop new models. These models are fundamental to recognition of objects and events in our world. Model of an object or event helps us in abstracting all sensory information into a simple symbol. The process of assigning a symbol to represent an object or event and then building more complex objects and events using these is at the heart of a field called *semantics*. Semantics is the study of meaning. Semantics is an important area of study in linguistics. Semantics is related to the study of meaning of words, phrases, sentences, and other larger units of text. Semantics is a rigorously studied field (see literature links at the end of this

[1] Note that this is a traditional classification of human senses. In later chapters we will see that humans have more senses, e.g.proprioception.

chapter). In our discussions in this book, we will not address detailed theory of semantics but we will consider the basic aspects as needed. For our purpose, we will just consider semantics to be the study of meaning associated with symbols. These symbols could be simple atomic symbols or could be composite symbols built by combining multiple atomic and/or composite symbols. Since our concern is with multimedia signals, these symbols could represent some units in different components such as audio and visual data (e.g., an area of a certain texture or an acoustic event) or could represent entities as combination of different media (such as an audio-visual object) thus resulting in symbols in multimedia .

Webster's dictionary defines experience as the "direct observation of or participation in events as a basis of knowledge." We experience the world we live in. The basis of all our interactions is our experience of the world. We learn about the world and accumulate and aggregate our experiences in the form of knowledge. Scientists among us *experi*ment to test their knowledge and to gain new knowledge. Scientific process relies on experiments to study a hypothesis under different conditions to evaluate its validity. Experimental aspects of a science are fundamental to its progress. Final evaluation of experiments is by humans using their sensory processes.

Communication is the process of sharing experiences with others. The history of civilization follows the development of our understanding of experiences and how to share them with our fellow humans even in other parts of the world immediately, as well as with those who will follow in future generations. It is interesting to see how many influential innovations and inventions in human history are related to communication and sharing of experiences with people who may be spatially and temporally separated. As explained in the previous chapter, this process started with the development of languages and has lead to the innovations resulting in the World Wide Web.

Information is an "efficient but abstract communication of experience." We gain knowledge through the set of experiences that make up our lives and communicate information about those experiences. Because we don't communicate the experiences themselves, we lose a vital element of the act of experiencing in the translation to information. Many of us can't know, for example, what it's like to hit the game's winning run, to surf the perfect wave, or to fight a war. We can read about the events, of course, and we can view pictures and videos. But we aren't present in these, so our experience and hence the knowledge of these events is incomplete.

In the communication process, one of the most important elements is to develop a dictionary. A dictionary is an exhaustive collection of a selection of the words of a language. It contains information about their meanings, pronunciations, etymologies, and inflected forms, in either the

same or another language. Thus, a dictionary is a shared and agreed collection of symbols (words) and what these symbols mean. Each language is based on a dictionary containing these symbols and rules, the grammar of the language, to use them. Without a dictionary communication may not happen. Just imagine situation when a person speaking English is talking to a person speaking Chinese. Each person is using a dictionary but these are two different dictionaries. Communication requires a shared dictionary. Dictionaries are commonly used in the context of languages and use words as the basic symbols as carrier of meaning. In computer science, dictionaries are extended to use list of codes, terms, and keys for use by computer programs. Many of the compression algorithms presented in later chapters put dictionaries into practice.

In multimedia, the basic carrier of meaning or symbols are not just traditional words as used in speech and text, but some units, similar to alphabet used in text and phonemes used in speech, in a particular medium. Let's consider visual information. Consider a very simple common task: Given an image of an object, name this object and list all objects that look like it and are related to it. Try to extend this to all detailed functions that you commonly see in a dictionary. Visual Dictionaries are being developed for different applications and for different kinds of visual information, ranging from shapes of curves to complex objects (see research articles). In these dictionaries, usually visual shapes or objects are stored and all their details are given in multiple languages. It is likely that these dictionaries will play increasingly important role in understanding of visual information and applications of emerging technology that will increasingly utilize cameras as information devices.

Objects and Events

In understanding data of any type, one tries to find which aspect of the world the data represents. As discussed above, perceptual processes depend on prior knowledge about the world we live in to analyze the signals. An important issue is how to represent the world.

Many researchers believed that the world can be represented using objects. This view believed that the world could be considered as a collection of objects. This view is challenged by many modern, and not so modern, thinkers (see literature at the end of this chapter). According to their views, events play equally important role. Events represent change in relationships among objects. And these changes are fundamental to understanding the current state of the world. According to emerging views, to model dynamic world both objects and events must be used. In a sense, objects are good in capturing static component of the world, while events complement that by capturing dynamic situations.

In computer science, object oriented thinking has been used in many fields and their applications. Object oriented approaches have revolutionized many areas of computer science because of the high level abstractions it offers for design, programming, and even some interactions.

Multimedia brings some new challenges to computer science, however. Multimedia, particularly audio and video, are fundamentally dynamic in nature. They capture signals that represent some attributes of the world as a function of time. In fact in many applications, even those sensors that capture static characteristics of the world, such as temperature at a point in space and time, are used to detect changes in those characteristics as function of time. A sensor is almost always placed in an environment where some event needs to be detected and the sensor measures some physical attribute that helps in detecting the event.

Many fields in computing have used the concept of event in designing systems. This concept has been used very differently in different fields, however. With increasing use of multimedia in computing, it is likely that a unified approach for event-based thinking will evolve (compare also research articles at the end of this chapter).

It must be emphasized that for modeling real world using powerful computational approaches, it is essential that both objects and events be used. Objects and events complement each other. Objects in computing capture attribute and functions of physical objects and other related concepts and events represent relationships and changes among those relationships in space and time.

Perception

Perception is the process of understanding sensory signals for recovering information (see literature). Perceptual processes have been analyzed with the goal to understand them for very long time. With arrival of computing, it attracted more attention from psychologists and researchers in artificial intelligence in the hope of developing machines for automatic perception. The understanding of perceptual processes has remained a difficult problem and is a very active research area in many disciplines including psychology, neuro-physics, and computer science. Understanding of sensory information is a very important step in many multimedia systems. We will study important perceptual processes in audio and visual processing in following chapters. Here we present some general aspects of perceptual processes.

A perception system takes sensory signals as input and generates the information that is relevant in its application context as output. This process is based on two important sources: signals and relevant knowledge sources. Figure 1 shows the role of existing knowledge in perception. The output of the system is clearly the combination of the input sensory signal as well as the knowl-

edge available to the system. Without any knowledge, the system can not produce any information, and without the signal the system can only hallucinate. Perception sometimes is considered as a controlled hallucination process (see research articles) where based on the signal the system starts hallucinating and creates multiple hypotheses then uses the signal to find the best support for its hypotheses and recovers information from signal.

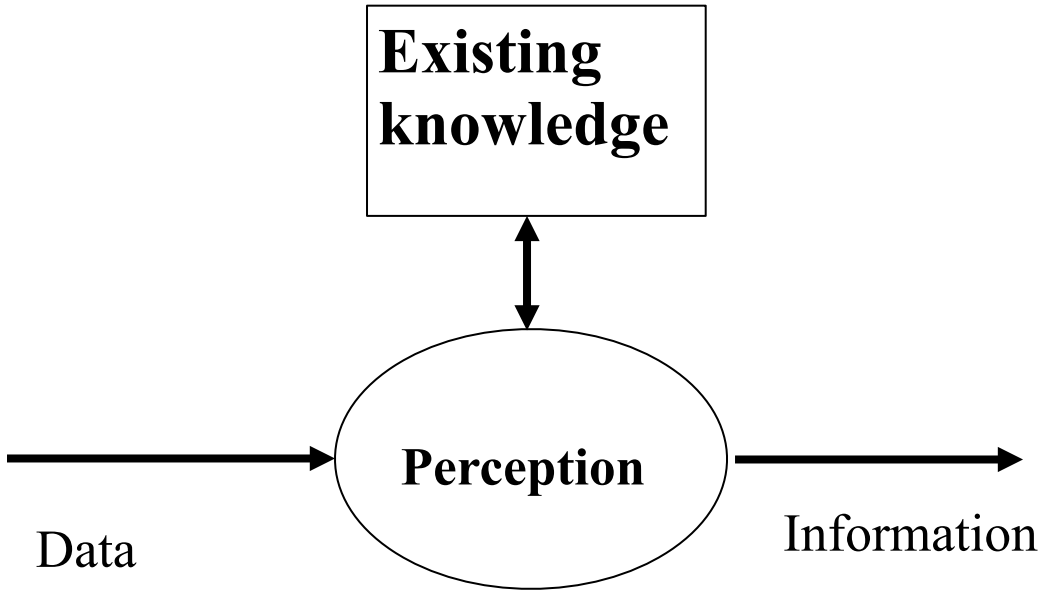


Figure 1. Perception is the process that recovers meaningful information from sensory data using existing knowledge.

A **B**

Figure 2: (A) A Dalmatian dog sniffing around. (B) Unstable perception: two faces or vase?

The role of knowledge, in the form of models of objects and events, is not immediately obvious. Some examples may make it very clear, however. We always use the term *recognition* for finding objects in signals, such as images. This term implies that we try to *re cognize* objects – meaning we know about the object or in other words have models of objects. Without models, there is no recognition. The models could be in many different forms ranging from very concrete and detailed model to very abstract models. To show the importance of models, we show two very commonly seen pictures in Figure 2. In Fig 2 a, there is a Dalmatian dog sniffing around. If you don't know how a Dalmatian dog looks, you will see only random blobs in this picture, but if you know Dalmatian dog, you will clearly see it. The Fig 2 b shows the picture which has two interpretations: you can either fit model of human faces to it and see two faces or see a vase in it. This shows that your perception system comes up with two hypotheses and admits both as viable, but only from a slightly different gaze point.

Perceptual Cycle

In all our activities, we use our senses, brain, and memory to understand our environment, to operate in this environment, to communicate about the environment and finally to build and update our knowledge repositories for efficient and effective use of what we learn. How we use these senses and how we convert this sensory data to information and knowledge has been a source of intrigue to thinkers for almost all known history. Here we present some ideas to provide historical context and perspectives on the evolution of this understanding.

Let's take a look at ancient philosophers (see literature references). They believed that:

- Understanding of world is indirectly derived using our senses.
- The fidelity of the model of the world depends on how well a person understands the world.
- People achieve different 'levels' of understanding in terms of their own knowledge structures.
- Nirvana is the highest stage of understanding.

These observations show deep insights about 2000 years ago. Thinkers even during that time clearly recognized that data from all sensors must be assimilated using existing knowledge to form an understanding of the environment. It was also recognized that sensors help us understand the world at different levels of understanding. One evolves to the highest level of understanding by refining their knowledge structures. Similar ideas and models are discussed by modern philosophers (see literature references) in theory of objective reality.

To formulate the problem from a computational perspective, we consider the so-called perceptual cycle introduced by Ulrich Neisser (literature and article references) in 1976. It attempts to model how people perceive the world. According to this model, a perceiver builds a model of the world by acquiring specific signals and information to accomplish a task in a natural environment. The perceiver continuously builds and refines a schema that is based on the signals he received so far. This schema represents the world as the perceiver sees it at that instant. The perceiver then decides to get more information to refine the schema for accomplishing the task that he has in mind. This sets up the perceptual cycle in Figure 3 below. The basic idea behind the perceptual cycle is that an agent is continuously interacting with the environment using its sensory mechanisms to build the model of the environment that will be useful in its task. At any given time instant it has a model of the environment, called schema that is constructed using all the data and information received until that point. The system then decides what more is required to complete the task and how that information could be acquired. Based on this, the agent collects more information using appropriate sensors.

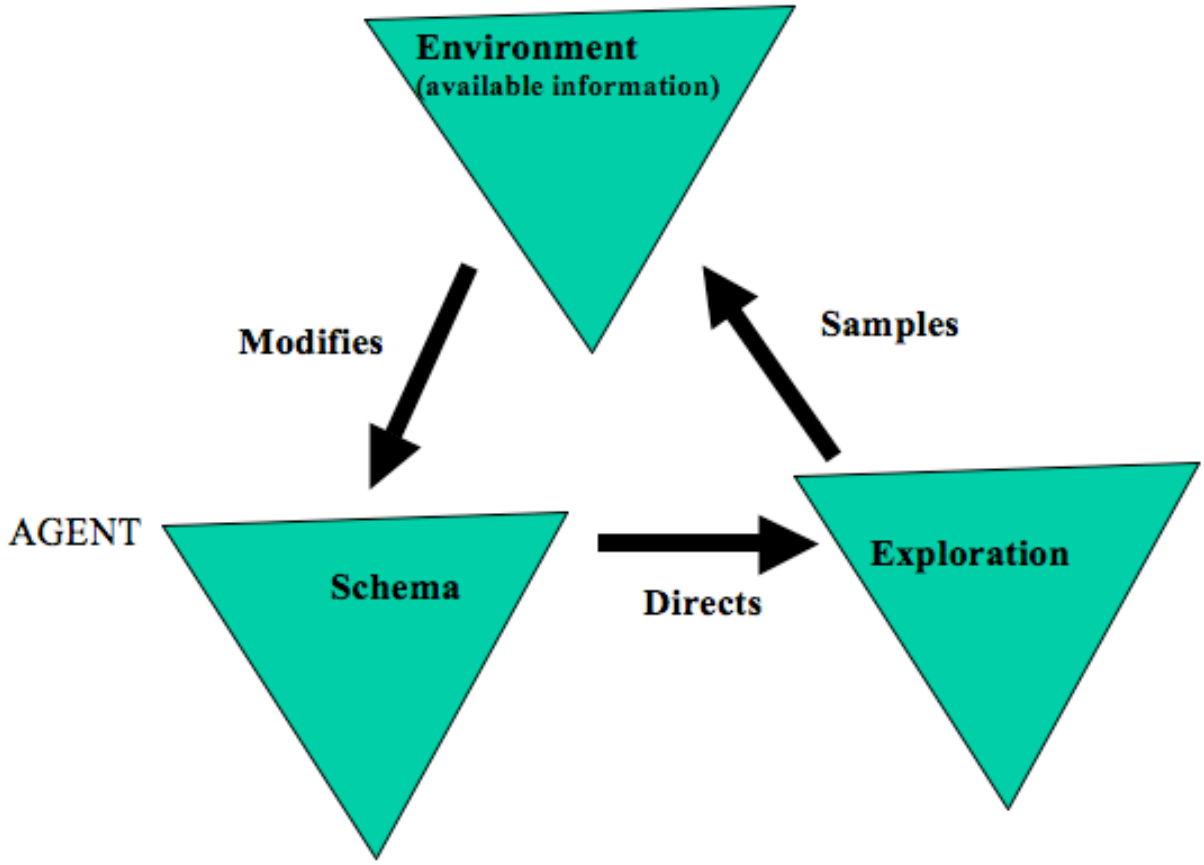


Figure 3. Neisser's perceptual cycle: The perceiver gets signals from the environment, interprets them using the current schema, uses the results to modify the schema, uses the schema to decide to get more information, and continues the cycle until the task is done.

The perceptual cycle model has conceptual similarity to recursive filtering techniques commonly used to estimate the state of a linear dynamic system using observers (sensors) that may provide noisy measurements. Chapter XXX will introduce these systems. The state of the system is represented mathematically as a vector. The state vector represents the values of the parameters that are used to characterize the linear dynamic system. In system theory (see research articles), these vectors represent the system so that correct amount of control inputs could be applied to the system to bring it into the desired state. In perceptual cycle, the schema represents the essential parameters that are required to solve the given task. Based on the current schema as compared to the final, or desirable, schema the agent must decide its action.

As mentioned, however, the perceptual cycle is dealing with perception that is not a linear dynamic system. This cannot be easily modeled using the current tools of the system theory. Some powerful estimators, such as Kalman filters (see research articles), have already been used in computer vision to model aspects of human perception. As progress in technology takes place, it

is expected that more formal tools will be developed to represent and construct schema using multimedia data.

Multimedia Systems

Consider a computing system equipped with multiple sensors working in a physical environment. The system continuously gets information about the environment from multiple sensors and must process all these in the context of its application. Obviously the applications could range from just identifying an object to autonomously functioning in a complex dynamic environment. Here we consider a general situation without any specific application. We also consider that for a computing system, the types of data sources are not limited just to the sensory modes that we humans can process. As is well known, different species have different sensory capabilities. Multimedia computing systems could be endowed with sensing capabilities of various types.

Let us assume that we are given $S_1, \ldots S_n$ data streams. These data streams have K types of data in the form of image sequence, audio stream, motion detector, annotations, symbolic streams, and any other type that may be relevant and available. We assume that these streams can be synchronized both in time and space resulting in these data streams represented in a common temporal and spatial coordinate system rather than in the coordinate system of each sensor. Further, we assume that metadata $M_1, \ldots, M_n$ for each stream is available from the original sources that helps us in interpreting the data stream in the context of the world. This meta data may include things like location and type of the sensor, viewpoint, angles, camera calibration parameters or any other similar parameters relevant to the data stream. Data stream is usually not directly very useful in the interpretation of the data in relation to the environment. In most cases, some feature detectors must be applied to each data stream to obtain features that are relevant in the current environment. Le us represent, features stream F_{ij} as the jth feature stream in S_i.

Given the above data environment, the most fundamental problem that multimedia computing systems must solve is to extract relevant information about the task at hand using data from these disparate sources. There are many challenging problems, including the following that are directly relevant to the main theme that we will address in this book:

- How do we combine these data streams to obtain the information that is essential for solving the problem at hand?
- How do we represent this data in the most compact form for communication and storage?
- How do we present this volume of data to a user in her computing environment to communicate intended information?
- What are the system issues that must be solved to deal with these disparate types of data and how they are handled by the system?

Before we address most specific concepts and techniques related to solving above problems in the rest of this book, some concepts that form the basic fabric of multimedia systems are discussed in the rest of this chapter.

Semantic Gap

Computing systems represent data in terms of bits and bytes and build from these more sophisticated representations such as lists, images, audio, and video. All these representations are fundamentally a collection of bits that programmers use to define abstractions to build their applications. On the other hand, the users of these systems are people who define their applications in terms of objects and events and build complex concepts based on abstractions that start with objects and events. There is a fundamental gap between the abstractions defined in computing systems and those used by the users of these systems. This situation is shown in Figure 4. This gap is defined in (Smeulders et al) as:

"*The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation. A linguistic description is almost always contextual, whereas an image may live by itself.*"

In current computers, we must build abstractions starting with bits, the most fundamental representation unit of data, and defining concepts that may be needed in specific applications. It is easy to build these concepts by defining various structures and naming and using them as a programmer may want to. We are all familiar with, and will discuss more in following chapters, concepts such as images, video, and audio signals as they are represented in computers. Human beings, currently usually ultimate users of the computing systems, usually do not think in these structures, however. Humans usually think of objects and events (see Quinton) and build complex concepts based on complex, often ill-defined and uncertain, relationships among them. As shown in Figure 4, there is a gap between the abstractions such as images and video as defined in computers and objects and events as used by people in their mind. This gap is what is called the semantic gap.

The main reason for the semantic gap, which often even exist among two persons, is that the physical world contains objects and events and people build the models of these objects and events in their mind based on the data that they receive from their sensors. People learn to abstract the data received from their sensors in terms of objects and events while combining this data naturally from all sources including all sensory organs, context, and memory. Chapter XXX

will explain some details of that. On the other hand, we try to define the models of these in computers using the data that is represented fundamentally in units of bits. The abstractions that are built in computers are based on what could be built in computers using bits. Fundamentally, in computing we define things based on what could be computed, while as humans we learn to abstract what is needed to survive.

Many concepts and techniques developed in multimedia computing are related to bridging the semantic gap. Starting from signal analysis approaches used in audio processing and computer vision to indexing for multimedia information retrieval, many concepts and techniques in multimedia address the problem of bridging the semantic gap. In fact, in all cases where human beings are integral part of a computing system, the semantic gap must be bridged. In many mature fields, this is bridged either by developing concepts in the field that bring data and humans conceptually close or by developing interfaces that rely on human intelligence to bridge the gap.

Consider common Internet search engines that appear to work so well that we use them every day: A close look at a search system's behavior shows that when searching for keywords that can be matched as character strings, it is easy to get good results. When you are searching for something that will not be satisfactory only based on string matching, but that requires some interpretation of either data or your intentions, search systems perform poorly. Most research in improving relevance of results in search engines is trying to bridge semantic gap. It is concerned with how to interpret data and how to detect a user's intentions based on contextual information.

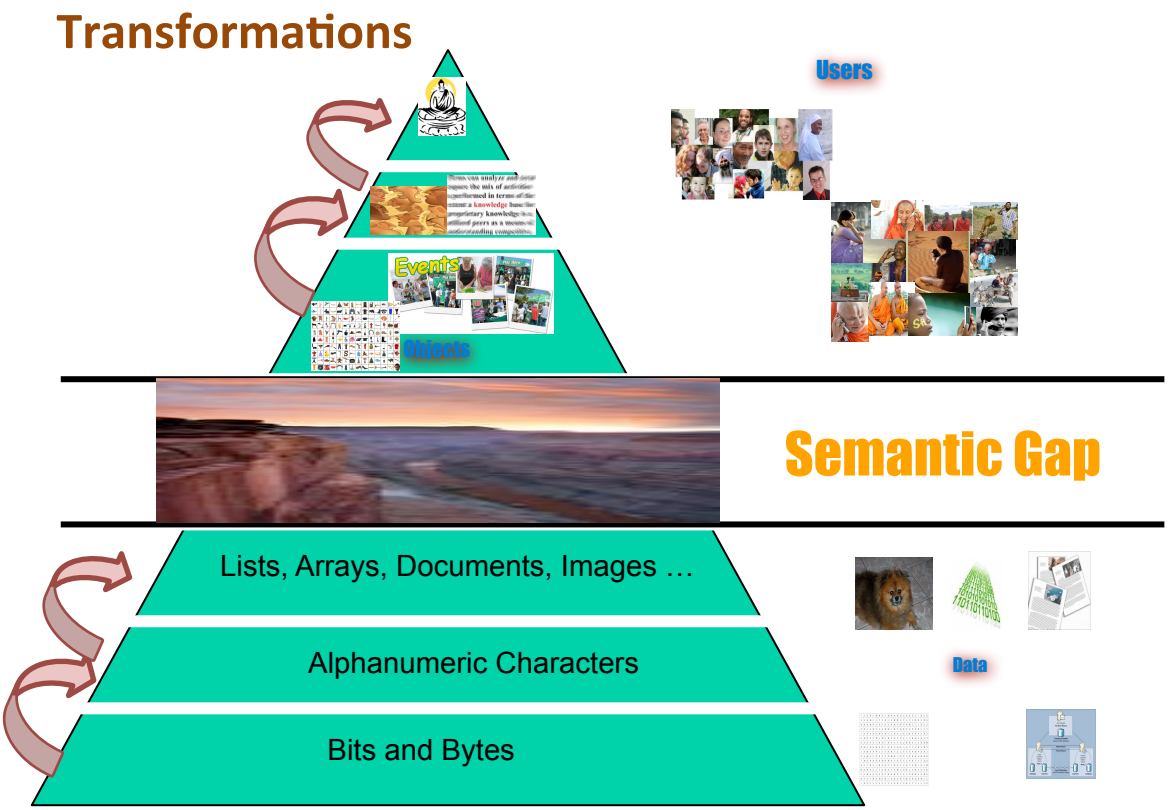


Figure 4: Semantic Gap. There is a big gap in how computers represent data like images in bits and bytes and how people think about images as collection of objects or events.

Metadata

Metadata literally means 'data about data'. Given some data that represents an audio, photo, or video; the metadata for it will describe different characteristics of this data. Some obvious metadata is name, size, date of creation, and type of the file. In addition to these, one can include any other information that is considered useful for understanding, storing, transmitting, presenting, or any other operation on the file containing the data. Metadata itself is data of some particular type; it is metadata in this particular context because it is used to qualify or help some other data.

Figure 5: This photo was taken at **Zhujiajiao, Shanghai , China.**

Since understanding techniques for text, audio, images, and other sensory data have not matured enough to correctly understand elements in data, metadata has gained in popularity particularly with the growth of the Web. XML was designed to be a mechanism to transport and store data. It accomplishes that by defining a language to describe what data is. Tags used in XML are data about data. Thus, XML is a language to associate metadata with the data explicitly so it could be read by any program. This helps in not only transporting and storing, but analyzing and understanding data.

Let's consider a picture file – say a photo shown in Figure 5. This photo was taken at Time (11:31 AM on Sept 13), Location (Zhujiajiao, Shanghai: Latitude 31 deg 6' 36.34" N, Longitude 121 deg 2' 59.22" E) and using a Nikon Coolpix P6000 camera. For this particular photo, no flash was used, and the focal length (6.7 mm), ISO (64), and aperture (f/4.7) values of the camera are known. All this (and much more) information is captured by the camera and is stored in the picture file along with the above pixel data using a popular EXIF (Exchangeable image file format) standard (compare web links).

In multimedia computing, use of metadata is increasing rapidly. Many approaches based on XML and tags are becoming commonplace even in audio, images, and video. It is expected that techniques to represent metadata as intimately as in text will evolve in this area. Use of EXIF with all stored digital images is a clear example of this trend in this field.

Context and Content

Content and context are two very commonly used terms in multimedia processing and understanding. There is no rigorous formal definition of content or context, though they are used ex-

tensively by practitioners and researchers. It is important to understand what they mean and how they are related to understand and develop efficient and effective multimedia systems.

Consider Figure 5 again. This is a photo containing 4224x3168 picture elements (pixels) and each pixel is a color pixel. This photo contains more than 13 Million points each with three color values associated with it. In the most basic form the content of the file or, as commonly used, photo are these 13M pixel colors in the spatial configuration as represented by the location of these pixels.

The metadata (discussed above), represents the context in which the data in the photo was acquired. Context is defined as the interrelated conditions in which some data (the content) is acquired. As seen above, some context parameters are stored by the camera using EXIF standards for the photo. EXIF standard is used almost by all digital camera manufacturers to store this kind of data with all digital photos.

Some other context parameters may help in understanding of data. For example, in the context of the above picture, EXIF tells the model of the camera but it will be very helpful if the owner of the camera is known and profile and calendar information about the owner is also available. In many cases based on this information it may be possible to understand what the objects are and more importantly, who the person in the picture is.

Multimedia research and techniques developed were concerned with only the content of the data in early days. Increasingly the importance of context is becoming clear. Recently (see Singavi) researchers are emphasizing that content and context should be combined and should be viewed as all information that must be used for understanding multimedia.

Index Terms

Experience, Perception, Meaning, Content, Context, Metadata, Perceptual Cycle, Semantic Gap, Neisser's Perceptual Cycle, Communication, Dictionary, Semiotic, Symbol, EXIF, Events, Objects.

Literature

1. L. M. Singavi, *That Which Is: Tattvartha Sutra (Sacred Literature),* AltaMira Press, 1998.
2. Hermann Kuhn, *The Key To The Center Of The Universe*, Crosswind Publishing, 2001.
3. Irvin Rock, *The Logic of Perception*, MIT Press, 1985.
4. Neisser, U (1976) Cognition and reality: principles and implications of cognitive psychology WH Freeman

Research Articles

1. D. E. Catlin, "Estimation, Control, and the Discrete Kalman Filter", Springer Verlag, 1988.
2. Igor A. Chimir, Waheeb A. Abu-Dawwas, Mark A. Horney, "Neisser's cycle of perception: formal representation and practical implementation", Journal of Computer Science (Special Issue): 106-111, 2005
3. Clowes, M. B. (1971), "On seeing things," *Artificial Intelligence*, Vol 2, No. 1, pp 79--112.
4. James J. Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1987.
5. Jonathan Metcalf, *Five Language Visual Dictionary,* 2003.
6. Jaroslav Peregrin *Meaning: The Dynamic Turn. Current Research in the Semantics/ Pragmatics Interface*. London: Elsevier, 2003.
7. Karl Popper, "Three World", The Tanner Lecture on Human Values Delivered by Karl Popper at The University of Michigan on April 7, 1978.
8. A Quinton, "Objects and events", Mind, pp. 197-214, 1979.
9. P. Sinha and R. Jain, "Semantics in Digital Photos: A Contenxtual Analysis", Int. Journal of Semantic Computing, Vol2, No. 3. pp 311-325, 2008.
10. Arnold Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain "Image Databases at the end of the Early Years" IEEE Transactions on Pattern Analysis and Machine Intelligence 23(1), January 2001.
11. A. Scherp and R. Jain, "An Eco System for Semantics," in IEEE Multimedia, June 2009.
12. G. Utz Westermann and Ramesh Jain," Towards a Common Event Model for Multimedia Applications", in IEEE Multimedia, January 2007.

Web Links

http://www.exif.org/

Exercises

1) Why does the semantic gap become a serious problem in perceptual systems?
2) A digital camera collects a lot of meta data related to camera parameters, including its location, and stores that with the intensity vales at every pixel. How can the meta data be used? Can this meta data help in reducing the semantic gap?
3) Where can EXIF be useful? Where can EXIF be harmful?
4) How is text related to audio?
5) Which is easier to analyze, speech or text? Discuss.
6) What is the role of knowledge in perception system? List at least 3 knowledge sources that could be used in understanding images?
7) What is a model as used in perception systems? Can you develop a recognition system without using a model?
8) How is the perceptual cycle related to estimation theory?
9) Multiple sensors are usually used to capture attributes of real world. Since the sensors have different coverage in space and have different temporal characteristics, how can one combine the data obtained from these sensors?
10) What is a feature in a sensor data stream? What role does it play in analysis of data and correlating different data sources?

PART II:
THE NATURE OF PERCEPTUAL INFORMATION

Chapter 4: Introduction to Sensors

Information about the environment is always obtained through sensors. Therefore, inorder to understand the nature of perceptual information, we must first start with an understanding on properties of sensors.

Properties of Sensors

In general, a sensor is a device that measures a physical quantity and converts it into a signal that an observer or instrument can read. Whether the sensor is human-made or from nature does not matter. An ideal sensor is sensitive to the measured property, insensitive to any other property, and does not influence the measured property. Of course, no perfect sensor exists because the laws of physics state that energy is conserved and sensors need a transfer of energy to function. After all, the photons absorbed in the retina of a human eye interfere with the universe in the sense that if they were not absorbed by the human eye, they would not be absorbed at all or would be absorbed by a different object (more on the properties of light later).

Fortunately, in practice, imperfect sensors are still useful. Not only that, but it's these deviations that multimedia computing uses for compression, corrects when reproducing signals, and analyzes for content. They shape the nature of the perceptual information we multimedia computing processes. Therefore it is very important to know typical sensor deviations, so we summarize them as follows before they are explained in more detail along concrete sensors in later chapters.

First, every sensor has a range and no sensor has unlimited range i.e., the possible intensities of the input signal lie within a certain interval. Going above that interval *saturates* the sensor i.e., whereas the ideal measurement response would suggest a further increase in output, the sensor outputs a maximum value and/or breaks (compare, for example, human ears exposed to too-loud noises). The range's lower bound is defined by the minimum amount of input that can be clearly distinguished from no input. If the output is not zero when the input is zero, the sensor is said to have an *offset* or *bias*. In practice, the sensitivity might differ from the measurement function specified.

Ideally, a sensor will respond linearly to the measured entity, i.e. a linear increase in input signal should result in a linear increase of the output of the sensor. A deviation from this, is often referred to as *nonlinear behavior*. Most sensors are tuned to behave linearly inside an *operational range*. The non-linearity is called *dynamic* when the sensor behaves differently based on time or other influencing factors that vary independently from the measured entity. A changing sensitiv-

ity given a constant signal is a *drift*. Most sensors have long-term drift due to aging. A random deviation from the measurement function is called *noise*.

The term h*ysteresis* refers to any deviation over time: When the measured entity reverses direction (for example, gets higher instead of lower) but the sensor's response has a finite lag, it might create one offset in one direction and a different offset in the other. Figure 1 illustrates the concept. A sensor's *resolution* is the smallest change it can detect in the quantity that it is measuring. Of course, resolution might also behave nonlinearly.

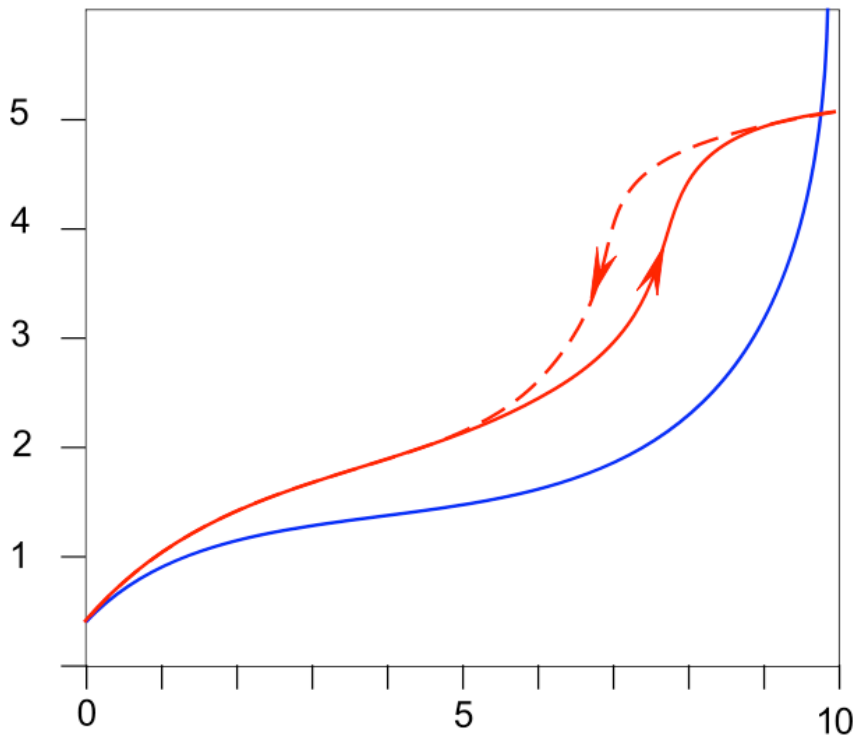


Figure 1. A general example for the hysteresis concept. The bottom curve (blue) is the sensor input, and the upper curve is the output. As the arrows show, the curve behaves differently when the measured entity decreases compared to when it increases.

Types of Sensors

Sensors for sound and light have been the most important for multimedia computing in the last decades because audio and video are best for communicating information for the tasks typically performed with a computer. That is, most people prefer to communicate through sound, and light serves illustrative purposes, supplementing the need for language-based description of a state of

the world. New or different tasks might use different sensors, however. For example, in real-world dating (as opposed to current implementations of online dating), communication probably occurs on many other levels, such as scent, touch, and taste.

Touch is seeing increased use in multimedia computing. Haptic technologies use tactile sensors to translate human motion into computation. Currently, the most common type of tactile sensor is force feedback, which is used in applications such as computer game controllers, servo mechanisms for aircrafts, and remote surgery. Haptic devices receive information from the user through pressure-sensitive sensors, such as piezo crystals, which emit electrical power proportional to the force applied to them. So-called *actuators* transmit the feedback using electro motors to induce a vibration or another physical motion to communicate to the human hands and arms. Other sensors, such as haptic gloves, allow normal usage of the hand but capture the muscular state of the fingers and other parts of the hand. Often, a remote robotic hand then reproduces the state of the operator's hand on the other end. A major issue with haptic sensors is tuning the feedback or sensor to make the user experience intuitive and natural, so the user does not have to learn how to operate the system.

In addition to light, sound, and touch, many animals have sensors for temperature, gravity, humidity, vibration, pressure, smell, and other properties of their environment. Many nerve systems and chemical sensors also sense internal aspects of the body. Some artificial sensors can sense physical as well as chemical phenomena, such as radiation, force, flow, and seismic activity. Although the principles described above apply, most of these sensors are not yet of interest in multimedia computing.

The following chapters will therefore focus on the properties of the two major environmental sensations: Sound and light. However, before that we introduce an important step, that usually follows right after the sensor in the processing chain: Digitization.

Digitization

In order to connect a sensor to a digital computer, the output of the sensor signal needs to be digitized. The electric current output by a sensor, and possibly amplified and/or integrated with other signals, is usually a continuous electrical signal, with the voltage directly proportional to the sound pressure. Digitizing is the representation of a signal by a discrete set of its samples, as Figure 2 shows. Instead of representing the signal by an electrical current proportional to its sound pressure, the signal is represented by on-off patterns representing sample values of the analog signal at certain fixed points. The on-off patterns are much less susceptible to distortions than analog signals. Copying in particular is usually lossless. Conceptually, digitization happens

in both the time and frequency dimension, illustrated in Figure 2. In a *sampling* step, the analog signal is captured at regular time intervals (the *sampling rate*), obtaining the signal's value at each interval. Each reading is called a *sample*. In *quantization*, samples are rounded to a fixed set of numbers (such as integers).

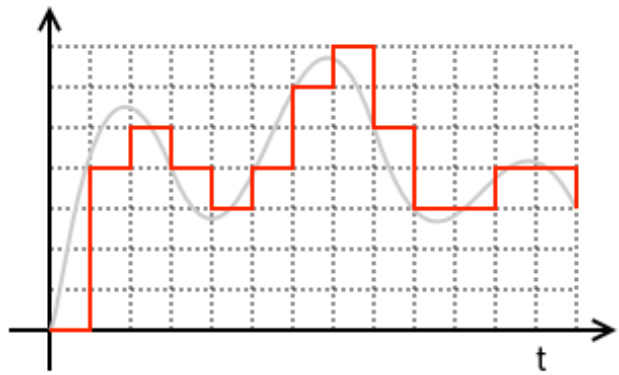


Figure 2. Digital representation of an analog signal. Both amplitude and time axis are discretized.

By generating the signal represented by each sample, we could transform a series of quantized samples back into an analog output that approximates the original analog representation. The sampling rate and the number of bits used to represent the sample values determine how close the reconstruction would be to the analog signal.

The error introduced by the quantization is the *quantization noise*. It affects how accurately the amplitude can be represented. If the samples have very few bits, the signal will only be represented coarsely, which will both affect the signal variance and introduce reconstruction artifacts. Typical bit representations for audio and video are 8, 16, 24, and 32 bits per sample.

The error introduced by the sampling rate is the *discretization error*. This error determines the maximum frequency that can be represented in the signal. This upper frequency limit is determined by the *Nyquist frequency*[2], which is half the sampling frequency of a discrete signal processing system. In other words, if a function $x(t)$ contains no frequencies higher than B hertz, $x(t)$ is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart.

To illustrate the necessity of $fs > 2B$, consider the sinusoid:

[2] Named after the Swedish-American engineer Harry Nyquist, or the Nyquist–Shannon sampling theorem.

$$x(t) = \cos(2\pi Bt + \theta) \;\equiv\; \cos(2\pi Bt)\cos(\theta) - \sin(2\pi Bt)\sin(\theta).$$

With $fs = 2B$ or equivalently $T = 1/(2B)$, the samples are given by

$$x(nT) = \cos(\pi n)\cos(\theta) - \underbrace{\sin(\pi n)}_{0}\sin(\theta) = \cos(\pi n)\cos(\theta).$$

These samples cannot be distinguished from the samples of

$$y(t) = \cos(2\pi Bt)\cos(\theta).$$

But, for any θ such that $\sin(\theta) \neq 0$, $x(t)$ and $y(t)$ have different amplitudes and a different phase, as Figure 3 illustrates.

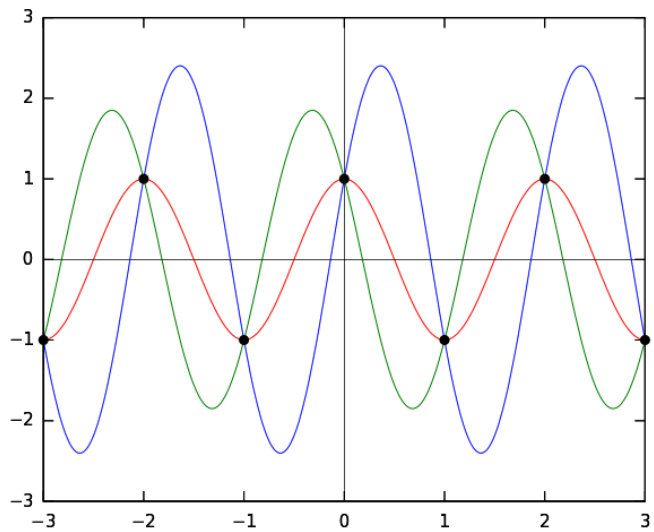


Figure 3. Three possible analog signals for the same sampling points.

Index Terms

Sensor, feedback, haptic, range, hysteresis, non-linearity, drift, saturation, offset, bias, actuator, habitation, digititization, quantization, sampling, discretization, Nyquist limit.

Research Articles

- A. Riul, Jr., R. R. Malmegrim, F. J. Fonseca, and L. H. C. Mattos: An artificial taste sensor based on conducting polymers, Biosensors and Bioelectronics, Volume 18, Issue 11, October 2003, Pages 1365-1369.
- Allen, J.F., “Maintaining Knowledge about Temporal Intervals,” Comm. of the ACM, November 1983, Vol. 26, No. 11, pp. 832-843.
- Krishna Persaud, George Dodd: Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose, Nature 299, 352 - 355 (23 September 1982).
- H. Nyquist. ”Certain topics in telegraph transmission theory,” *Trans. AIEE*, vol. 47, pp. 617-644, Apr. 1928. Reprint available as classic paper in: *Proceedings of the IEEE*, Vol. 90, No. 2, Feb 2002.

Exercises

1. Explain how an external observer would perceive sensor hysteresis.
2. Traditional wisdom gives humans five senses: vision, hearing, touch, smell, and taste. Explain why this number is wrong and discuss why it is not easy to define what a sense is.
1. Habituation describes an effect in which repeated exposure to a stimulus leads to lower response. Provide examples of human sensors that show habituation.
2. Discuss and experiment with ideas to reconstruct frequencies beyond the Nyquist limit. What are the trade-offs?
3. Explain how the Nyquist limit sometimes becomes visible in the image and video domain. What are the typical artifacts?
4. Explain why haptic sensors in virtual reality require physical robustness beyond what is often technically achievable.

Chapter 4: Sound and Hearing

Hearing and vision are the two most important sensual inputs that humans can process. Many parallels exist between visual signal processing and acoustic signal processing, but sound has unique properties—often complementary to those of visual signals. Perhaps this is why nature gave animals both visual and acoustic sensors: To gather mutually additive information about the environment. Many species use sound to detect danger, navigate, locate prey, and communicate. Earth's atmosphere and water, as well as virtually all physical phenomena—fire, rain, wind, surf, earthquake, and so on—produce unique sounds. Species of all kinds, such as frogs, birds, marine and terrestrial mammals, have developed special organs to produce sound. In some species, these have evolved to produce singing and speech. Furthermore, humans have developed culture and technology (for example music, telephone, and radio) that let them generate, record, transmit, and broadcast sound.

In this chapter, we introduce the basic properties of sound, sound production, and sound perception. Of course, a multimedia book can only scratch the surface of this complex and fascinating topic.

The Physics of Sounds

The *American Heritage Dictionary of the English Language*, Fourth Edition defines sound as "a traveling wave which is an oscillation of pressure transmitted through a solid, liquid, or gas, composed of frequencies within the range of hearing and of a level sufficiently strong to be heard, or the sensation stimulated in organs of hearing by such vibrations." This compressed formulation is perfect start for discussing the properties of sound. Sound is generated by mechanical oscillation. Unlike light, sound must travel through a medium. In a vacuum there is no sound so, for example, one can't hear exploding space ships. The traveling speed of sound varies according to the medium it is traveling in. In dry air at 20 °C, the speed of sound is 343 meters per second, or Mach 1. In fresh water, also at 20 °C, the speed of sound is approximately 1,482 meters per second.

Sounds being vibrations, their waves are sinosoidal in nature, i.e. composed of sine waves. For a sound to be heard, the oscillation's frequency and amplitude must be in a certain range. For humans, hearing is normally limited to frequencies between 12 and 20,000 Hz (20 kHz). The upper

limit generally decreases as the person ages. Other species have a different range of hearing. Dogs, for example, can perceive vibrations higher than 20 kHz[3].

Because sound is an oscillation of pressure, you can measure a sound wave's amplitude by measuring the *sound pressure*. Sound pressure is defined as the difference between the average local pressure of the medium outside of the sound wave it is traveling through (at a given point and a given time) and its pressure within the sound wave. The square of this difference is usually averaged over time and/or space. By taking a square root of the average, you obtain a root mean square (RMS) value.

The sound pressure perceived by the human ear is nonlinear (see also Chapter XXX [mulaw], where this property is exploited for compression), and the range of amplitudes is rather wide. Therefore, sound pressure is often measured as a level on a logarithmic scale, the *decibel scale*. The sound pressure level (SPL) or L_p is defined as:

$$L_p = 10 \log_{10} \left(\frac{p^2}{{p_{ref}}^2} \right) = 20 \log_{10} \left(\frac{p}{p_{ref}} \right) \text{ dB}$$

where p is the RMS sound pressure and p_{ref} is a reference sound pressure. Commonly used reference sound pressures for silence, defined in the standard ANSI S1.1-1994, are 20 μPa (micro-Pascal) in air and 1 μPa in water. Most sound recording equipment is calibrated to omit 0-amplitude at these levels.

The human ear does not have a flat spectral response (we discuss this in more detail later in the chapter). That is, the same sound pressure at a different frequency will be perceived as a different volume level. Therefore, sound pressures are often frequency weighted so the measured level will more closely match perceived levels. The *A-weighting scheme*, defined by the so-called I*nternational Electrotechnical Commission (IEC)* and illustrated in Figure 1, is the most common.

[3] This is also one reason why dogs and cats will not react to broadcast TV the same way humans do. Even though the squeak of a mouse can draw a cat's attention from hundreds of meters away, the same sound, much more intense, from an MP3 player or a TV might not interest the cat at all.

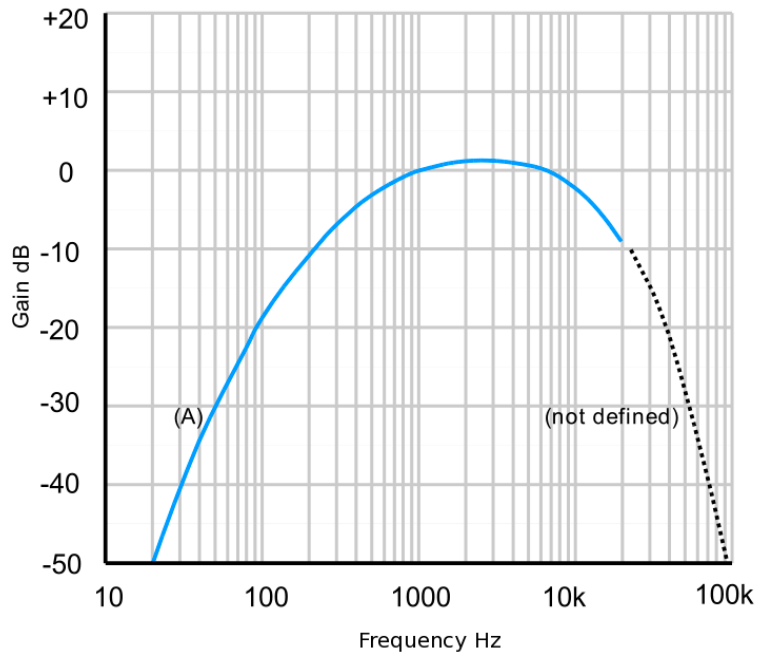


Figure 1. Sound pressure A-weighting scheme according to IEC 61672:2003.

Sound pressure levels weighted by this scheme are usually labeled as dBA or dB(A). The terms dB and dBA, like the percent symbol (%), define ratios rather than physical measurement units. A value of 10 dB can refer to completely different sound pressure levels, depending on the reference.

Observed Properties of Sound

In practice, sounds do not travel exclusively in a homogenous medium from a source to exhaustion. The environment is full of objects that sound can travel through or be reflected from. Sound pressure waves can collide with each other. The resulting effects of these conditions play a large role in the design of multimedia systems. The three most important sound effects are echo, reverberation, and interference.

An *echo* is a reflection of sound, arriving at the listener some time after the original sound. Typical examples are the echo produced by the bottom of a well, by a building, or by the walls of a closed-in room. Because most materials easily reflect sounds, echoes are always present in every environment. The time delay is the extra distance divided by the speed of sound. The human ear cannot distinguish an echo from the original sound if the delay is less than 1/10 of a second. Thus, because the velocity of sound is approximately 343 meters per second at a normal room

temperature of about 20°C, the reflecting object must be more than 16.2 meters from the sound source at this temperature for a person at the source to hear an echo. Physics tells us, however, that traveling requires energy. Of course this is also true for sound waves (otherwise we would be swamped with all sounds and vibrations happening in the Universe). Therefore, for a sound wave to travel that far back and forth it must have sufficient energy. Normal conversation, for example, is usually below this energy threshold and no echo is perceived.

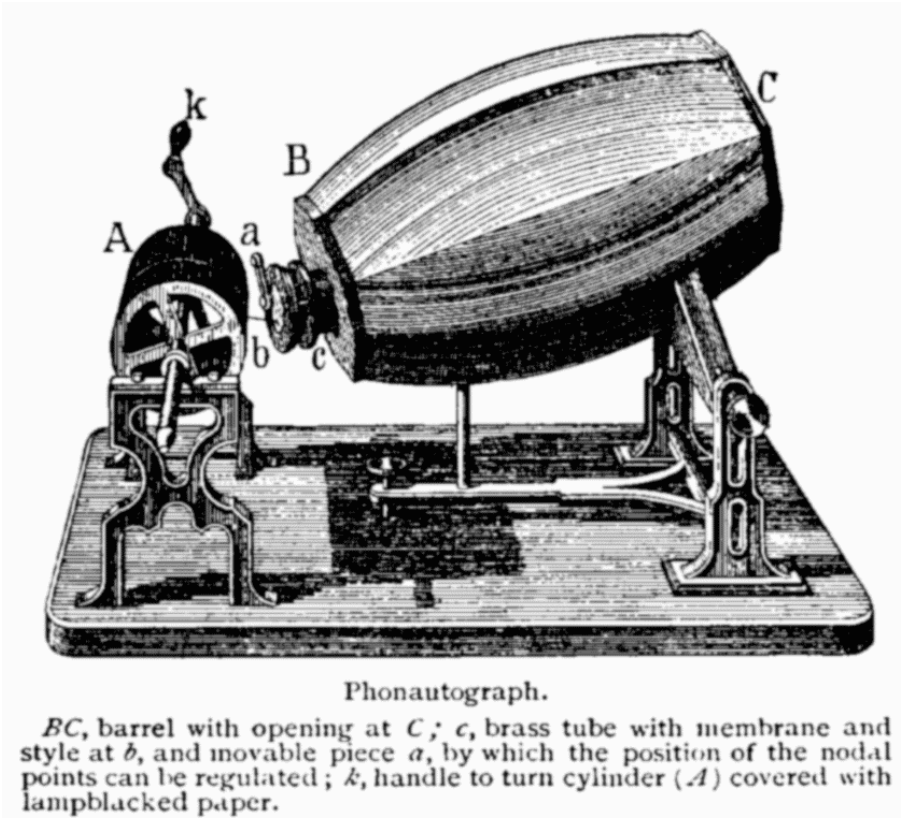


Figure 2. Phonautograph by Édouard-Léon Scott de Martinville. (Source: Uncredited 19th century engraving, Wikimedia Commons)

A *reverberation* is created when a sound is produced in an enclosed space, causing numerous echoes to build up and then slowly decay as the environment absorbs the sound. This is most noticeable to the human ear when the sound source stops but the reflections continue, decreasing in amplitude, until they are no longer audible. Reverberation receives special consideration in the architectural design of large chambers, which need specific reverberation times to achieve optimum performance for their intended activity. Unless a room and recording equipment is specially designed to not cause reverberation, reverberation is always present. Reverberation is also present during the production of speech in the vocal tract. Reverberation is characterized by the *re-*

verberation time, that is, the length of this sound decay. Multimedia content analysis techniques often suffer from not accounting for reverberation, even when it is inaudible.

Interference is the superposition of two or more waves that results in a new wave pattern. It usually refers to the interaction of waves that are correlated or coherent with each other, either because they have the same source or the same or nearly the same frequency. Sound interference causes different effects, which are described in wave propagation equations in physics.

Multimedia system designers should be aware of interference, which they can use constructively and destructively.

Consider two waves that are in phase, with amplitudes A_1 and A_2. Their troughs and peaks line up and the resultant wave will have amplitude $A = A_1 + A_2$. This is known as *constructive interference*.

If the two waves are 180° out of phase, one wave's crests will coincide with another wave's troughs and so will tend to cancel each other out. The resultant amplitude is $A = |A_1 - A_2|$. If $A_1 = A_2$, the resultant amplitude will be zero. This is known as *destructive interference*. Audio engineers and signal processing experts often use destructive interference to eliminate unwanted sounds—for example, in noise-canceling earphones. However, this is not the same as masking effects, which are caused by the perceptual properties of the human brain (and will be discussed in Chapter XXX).

Recording and Reproduction of Sound

Humans have tried to accurately record sound for a long time. The first notable device that could record sound mechanically (but could not play it back) was the phonautograph, developed in 1857 by Parisian inventor Édouard-Léon Scott de Martinville. This machine produced *phonautograms*, the earliest known recordings of the human voice. These earliest known recordings include a dramatic reading in French of Shakespeare's Othello and music played on a guitar and trumpet.

Figure 2 shows a schematic of the device from the inventor's original records. A barrel with an opening captured the sound waves and focused them onto a membrane attached to a hog's bristle , causing the bristle to move and allowing it to inscribe the sound onto a visual medium. Even though this device was more an early oscillograph than a sound recording device, the concept of the first practical sound recording and reproduction device wasn't too different.

Thomas A. Edison invented the mechanical phonograph cylinder in 1877 and patented it in 1878. The recordings were initially stored on the outside surface of a strip of tinfoil wrapped around a rotating metal cylinder. To play back the recordings, a needle ran along the cylinder, applying less pressure than in the recording to convert the mechanical engravings into sound waves that would be mechanically amplified. Figure 3 shows a US postage stamp featuring the device.

Figure 3. Edison's phonograph on a US postage stamp.

Not surprisingly, sound recording still obeys the same principles, with two main exceptions. The sound waves are converted to electrical waves by a microphone, and recorded to a medium. Most of today's storage media is digital—that is, sound waves are converted to binary numbers before they are imprinted on the medium. The media themselves, such as CDROM or DAT, are a bit more sophisticated than Edison's cylinders. Currently, generic media, such as hard disks and flash memory, are replacing these specialized media.

The next sections explain the governing principles of modern sound processing.

Microphones

A microphone is an acoustic sensor that converts sound into an electrical signal. The general principle is that sound pressure is inflicted on a membrane that varies its electrical resistance according to the movement. Most current microphones use electromagnetic induction (dynamic microphone) by letting the membrane swing a magnetic field produced by a coil, capacitance change (condenser microphone) by letting the membrane be part of a capacitor that varies capacity with movement, or piezoelectric generation (piezo crystals emit electricity when under pres-

sure). Modern laser microphones use light modulation to produce the electric signal by "watching" the mechanical vibration.

A single dynamic membrane will not respond linearly to all audio frequencies. For this reason some microphones use multiple membranes for the different parts of the audio spectrum and then combine the resulting signals. The different microphone types have different electrical properties. A complete microphone also includes a housing and a means of bringing the signal from the element to other equipment (such as wires or RF capability). These and other characteristics of the microphone, such as diaphragm size, intended use, or orientation of the principal sound input to the principal axis (end- or side-address), determine the properties of the recorded sound space. This, when planning a recording, it is best to survey what is available in the market and read vendor specifications.

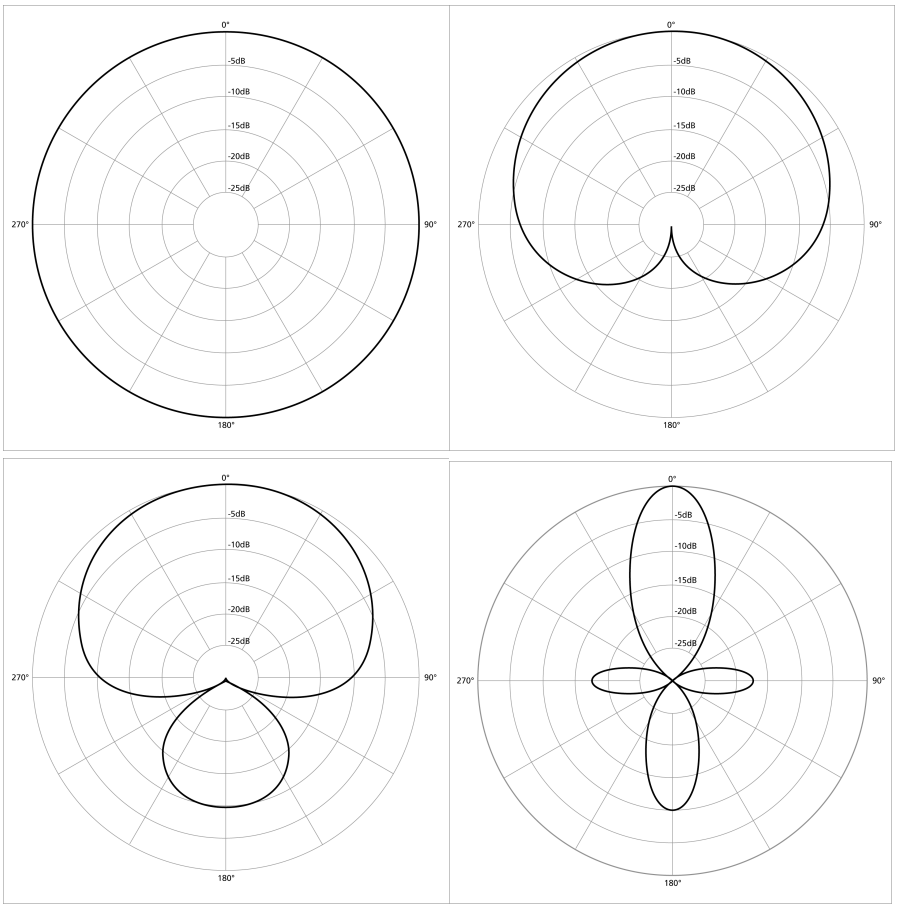


Figure 4: Four common polar patterns of microphones: (top left) omnidirectional, (top right) cardiod, (bottom left) supercardiod, and (bottom right) shotgun (images from Wikimedia Commons). The diagrams are created by recording a constant sound (frequency,

intensity) from different directions and plotting the microphone response in each directions. Note, that the resulting graphs are usually frequency-dependent, therefore often an averaged result is shown.

A microphone's most important characteristic is its directionality, which indicates how sensitive it is to sounds arriving at different angles around its central axis. A microphone's directionality is usually represented by a polar pattern showing the location of points that produce the same signal level output in the microphone if a constant sound pressure level is generated from that point.

Figure 4 shows some idealized example patterns. The patterns are considered idealized because in the real world, polar patterns are a function of frequency. Manufacturer diagrams therefore usually include multiple plots at different frequencies. Also, although an omnidirectional microphone's response is generally considered to be a perfect sphere in three dimensions, in the real world this is not the case. The microphone's body is not infinitely small and, as a consequence, it tends to get "in its own way" with respect to sound arriving from the rear, causing a slight flattening of the polar response. This flattening increases as the microphone's diameter (assuming it's cylindrical) reaches the wavelength of the frequency in question. Therefore, the smallest-diameter microphone will give the best omnidirectional characteristics, especially at high frequencies. Different microphone properties result in different applications.

Headset and *lavalier* microphones are made for hands-free operation. These are small microphones worn directly on the body. Originally, they were held in place with a lanyard worn around the neck; now, they are typically fastened to clothing with a clip, pin, tape, or magnet. These directed microphones allow mobile use for voice recording. They are in everyday use for videoconferencing, personal recording, theatrical performances, and dictation applications.

A *parabolic* microphone uses a parabolic reflector to collect and focus sound waves onto a microphone receiver, very similar to a satellite dish. Typical uses of this microphone, which has unusually focused front sensitivity and can pick up sounds from many meters away, include nature recording, outdoor sporting event recording, and eavesdropping. Because these microphones tend to have poor low-frequency response as a side effect of their design, they are not typically used for standard recording applications. However, machine intelligence might be able to infer information from them (for example, in connection with a surveillance camera).

Noise-canceling microphones have a highly directional design intended for noisy environments when direct attachment to the body is not desirable. For example, a vocalist might use this type of microphone on a loud concert stage. Often, noise-canceling microphones combine signals received from two membranes that are in opposite electrical polarity or are processed electroni-

cally later. The main membrane is mounted closest to the intended source and the second is positioned farther away from the source so that it can pick up environmental sounds to be subtracted from the main signal by destructive interference.

Arrays of *omnidirectional* microphones are best to pick up as much sound from the environment as possible. These are typically used for auditory scene analysis, where objects can be located by analyzing the time delay of arrival between microphones (due to the speed of sound). In addition, combining the signals from a larger set of microphones can enhance the signal quality. This technique is often used in speech recognition, when head or body-mounted microphones are not desirable.

The electric current output by a microphone, and possibly amplified and/or integrated with other signals (mixed) by further equipment, is a continuous electrical signal, with the voltage directly proportional to the sound pressure. In practice, sound recording has a linear area for certain sound pressure levels and frequency ranges and nonlinear areas if the sound pressure and/or the captured frequency is too low or too high. If the sound pressure level is too low, the signal will mostly just be zero; if it is too high, it will reach an internal clipping point (which in the worst case is a short circuit) and will be severely distorted. Even if it does not reach the clipping point, the nonlinear behavior of sound processing devices will lead to distortion when the captured signal is outside the nonlinear scope. This is referred to as the signal being *overdriven*. When the signal is outside the recording device's linear frequency range, harmonic distortion is introduced. For example, a sine curve might be converted into a much less regularly shaped signal. Figure 5 shows an example.

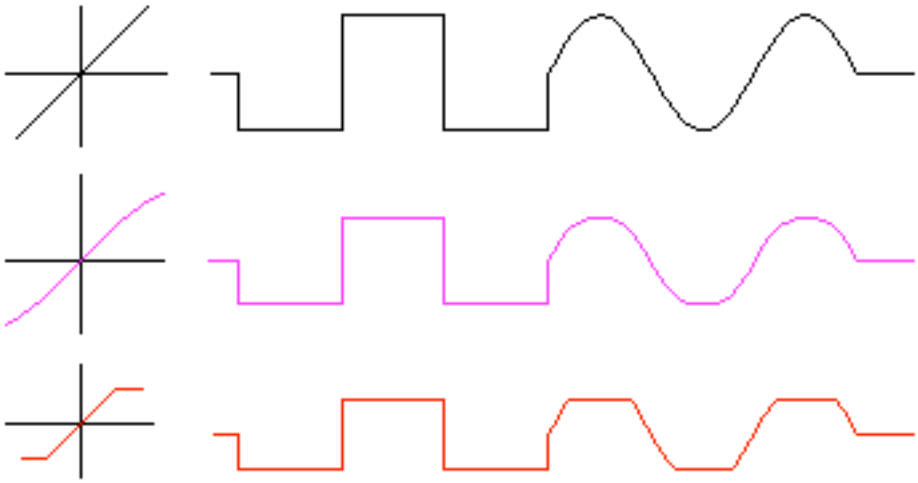


Figure 5. The diagrams on the left show the behavior of an amplifier, the curves on the right show the results for different input signal shapes. Top: Linear behavior, original sig-

nal, second row: typical analog behavior and distorted signal due to overdrive, bottom: typical digitization behavior and distorted signal from clipping.

A microphone's output is usually amplified using an analog amplifier before it is digitized. Some microphones already output a digital signal directly as standardized by the so-called AES 42 standard.

Even if standardized, all sound processing devices have slightly different linear ranges. Sound cables, especially when very long, can also inhibit certain frequencies and, because they often work as "involuntary antennas," might introduce electric distortion from the outside, the most current one being a "buzz" from the 50 Hz/60 Hz electrical system. Then, recording media, such as old vinyl records or audio cassettes, introduce their own nonlinearities and the effects stack up with each copy made. Therefore, in the last two decades, sound processing has shifted from analog to digital. At the time of this writing, many microphones, mixers, and pre-amplifiers are still analog, but storage and processing is digital. With standards such as AES 42 growing increasingly popular, digitization will soon become a much earlier part of the processing chain.

Due to the Nyquist limit and because the maximum frequency perceptible by the human auditory system is about 22 kHz, compact discs sample at 44 kHz. Human speech, which usually peaks between 6 and 8 kHz, is considered completely represented by a 16-kHz sampling frequency. Professional audio recording equipment frequently use sampling frequencies about 44 kHz, such as 48 kHz and 96 kHz. If the analog equipment supports it, these devices can capture frequencies that are imperceptible by the human ear, allowing for better reproduction of overtones. Further processing, such as digital filters and machine learning, might use the higher frequencies too .

Reproduction of Sound

Up to this point, we have assumed the existence of a sound signal. This assumption is generally a safe one, because sound pressure levels can be measured virtually anywhere on earth. However, reproducing sound from storage requires special devices. The most common sound reproduction device is the loudspeaker.

A loudspeaker (or speaker) is the exact reverse of a microphone. A speaker is an electroacoustic transducer that converts an electrical signal into sound. The speaker pulses according to the variations of an electrical signal and causes sound waves to propagate through a medium such as air or water. A speaker usually consists of a membrane that is driven back and forth and made to oscillate using an electrical-to-mechanical force converter, such as an electromagnet or a piezo crystal. This core part is typically called the *driver*. The term "loudspeaker" can therefore refer to

individual drivers or to an integrated system of drivers in an enclosure. The role of the enclosure, apart from providing a place to mount the drivers, is to prevent sound waves emanating from the back of a driver from interfering destructively (that is, by causing cancellation) with those from the front.

To adequately reproduce a wide range of frequencies, most speakers require a combination of drivers with different properties. Each individual driver is then used to reproduce a different frequency range. Common driver types include subwoofers (very low frequencies, typically below 120 Hz), woofers (low frequencies), mid-range speakers (middle frequencies), tweeters (high frequencies), and sometimes supertweeters, which are optimized for the highest audible frequencies. In systems using multiple drivers, a network of electrical filters, called a *crossover*, separates the incoming signal into different frequency ranges and routes them to the appropriate driver. A speaker system with n separate frequency bands is called an n-way speaker. Typical home audio devices have a three-way speaker system, consisting of a woofer, a mid-range speaker, and a tweeter.

Like microphones, speakers have directionality—that is, their frequency (re)production properties vary in space. Figure 8 shows the directionality of a typical column-shaper home audio system speaker.

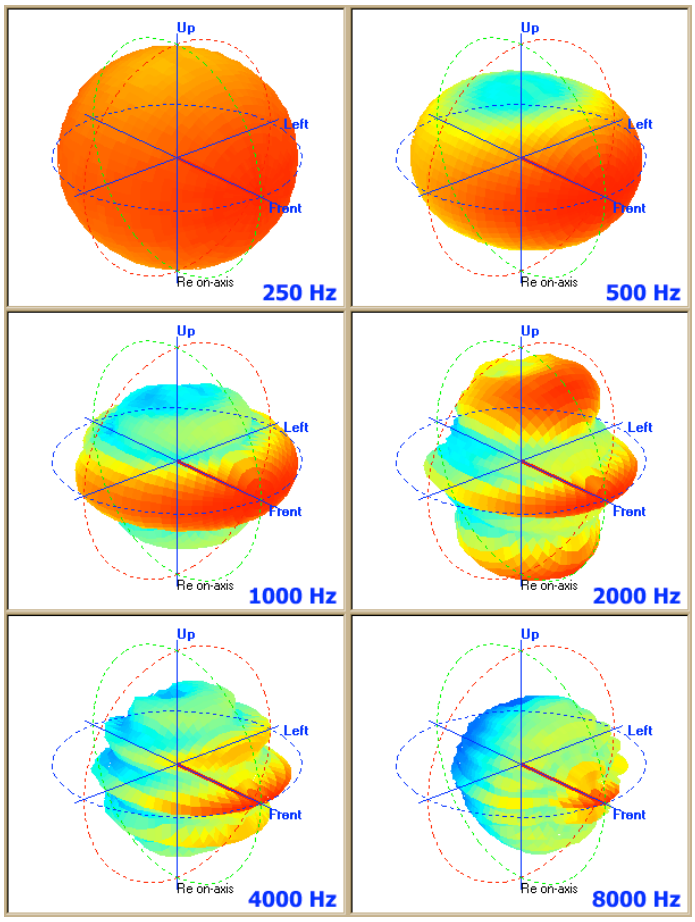


Figure 8. Polar patterns of a typical home speaker system consisting of four drivers at different frequencies. (Source: Wikimedia Commons)

Needless to say, speakers are designed with different directionality for different applications (for example, car speakers have a different polar pattern than supermarket speakers used for making announcements). Other factors that determine a speaker's properties are

- the rated power, which determines the maximum input a speaker can take before it is destroyed;
- the maximum sound pressure level (SPL), which defines how much sound pressure the speaker can emit;
- the impedance, which determines the electrical compatibility with different amplifiers;
- the crossover frequencies, which define the nominal frequency boundaries of the signal division by the drivers; and
- the frequency range, which determines the speaker system's linear frequency response range.

The enclosure type determines some of the loudspeakers' perceptual properties. Another important factor for sound reproduction quality is the relationship between the number of channels

used (for example, two four, or six), how they have been encoded (for example, stereo or surround), and how the speakers are placed in the room when reproducing sound.

Speakers and microphones are the most variable elements in terms of perceived sound quality. Except for lossy compression, they are responsible for most of the distortion and audible differences in sound systems. Our practical advice to the reader is to use high-quality headphones when experimenting with sound algorithms.

Index Terms

sound,

Exercises

1. How many dBs are 50%, 1%, 0.01%, and 200%? How many dBs can be stored in 16 bits, 24 bits, and 32 bits?
2. List the factors that would influence echo and reverberation in a lecture hall.
3. You are a researcher on a project that requires you to make frequent sound recordings. Unfortunately, your officemate needs to have a very noisy server farm standing beside him. Given no social rules or limitations, what would be the best thing to do to isolate the noise?
4. Discuss what would be the best directionality for a microphone that is used for field studies where you interview people in noisy environments.
5. Explain the artifacts you would expect from a microphone/loudspeaker that is forced to record/play sound both (a) outside its frequency range, and (b) outside its amplitude range.
6. Assume you want to record a classroom seminar with many participants. What environmental noise would you expect?
7. When a signal is received by a microphone, it is amplified and passed out of a loudspeaker. The sound from the loudspeaker might be received by the microphone again, amplified further, and then passed out through the loudspeaker again. The effect is known as *Larsen effect* or, more colloquially, as the *feedback loop*. Describe what happens and what the signal looks like.

Literature

Data Interchange on Read-only 120 mm Optical Data Disks (CD-ROM). ECMA-130. June 1996.

F. Alton Everst, Ken Pohlmann. *Master Handbook of Acoustics*, McGraw-Hill, 5th edition, June 2009.

D. T. Blackstock. *Fundamentals of Physical Acoustics*, Wiley-Interscience, 1st edition, February 2000.

B. Gold, N. Morgan. *Speech And Audio Signal Processing: Processing And Perception Of Speech And Music*, John Wiley & Sons, 1st edition, August 2006.

Chapter 5: Production of Speech and Music

Sound creation tools are very sophisticated and create signals with unique and interesting properties on the signal level. These properties are widely exploited on many levels in multimedia computing when compressing, editing, and analyzing sounds. We therefore did dedicate this short Chapter to it where we summarize the most salient properties of signals coming out of music and speech production. Needless to say, it would be easy to write a book about even only a single musical instrument, therefore this chapter can only convey a top-down view from the tip of the iceberg (with many many details staying unnoticed below the water line).

Production of Speech

The production of random noise is relatively easy, but modulating sound in a way suitable for communication requires sophisticated apparatus. Although humans are not the only species to produce sophisticated sounds, they seem to have developed the most sophisticated expressiveness.

The frequency range of speech is between 80 Hz and about 5 kHz. The pitch of the human voice is between 120 and 160 Hz for adult males and between 220 and 330 Hz for women and children. Vowels can reach frequencies up to about 5 kHz. Sibilants emit the highest frequencies, which can easily reach the nonaudible spectrum (above 20 kHz). The frequency dynamics of speech are relatively high compared to many other sound sources, such as some musical instruments. In general, the volume of the human voice is limited to the sound energy the human body can produce. At 60 centimeters from the mouth, the human voice can typically reach a sound volume of about 60 dBA. A stronger voice can raise the volume by about 6 dB. Yelling measures about 76 dBA (males) and 68 dBA (females).

The articulatory phonetics field, a branch of linguistics, investigates how sounds are produced by the tongue, lips, jaw, and other speech organs. As already explained, sound is pressure waves traveling through air. Therefore, human speech is directly connected to the body's respiratory system. Almost all speech organs have additional functions. In addition, of course, different speech organs have more than one function in speech production, and, to make matters even more complex, the same sounds can be produced by different combinations of organs. Figure 1 shows a schematic of the human speech production apparatus.

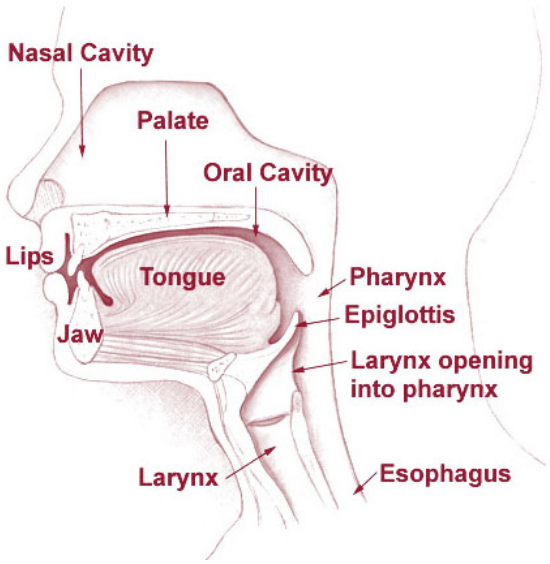


Figure 1. A schematic of the human speech production apparatus. (Source: National Cancer Institute)

Speech sounds are usually classified as

- *stop consonants* (with blocked airflow, such as the English pronunciation of "p," "t," or "k"),
- *fricative consonants* (with partially blocked and therefore strongly turbulent airflow, such as the English "f" or "v"),
- *approximants* (with only slight turbulence, such as the English "w" and "r"), and
- *vowels* (with full unimpeded airflow, such as the English "a," "e," "i," and "o").

Other classes exist and there is variation among languages, in Mandarin for example tonality determines meaning. Vowels are usually classified into *monophtongs*, having a single vowel quality; and *diphtongs*, which manifest a clear change in quality from start to end as in the words bite, bate, or boat.

Consonants and vowels are the building blocks of speech. Linguists refer to these building blocks as *phonemes*. American English has 41 phonemes, although the number varies according to the speaker's dialect and the system that the linguist doing the classification uses. A phoneme's concrete pronunciation depends on the previous and the following uttered speech sounds. It also depends on the type of speech (for example, whispering versus screaming) and the speaker's emotional state, as well as the anatomy of the throat, age, native language and dialect, and social and environmental surroundings. Diseases of the lungs or the vocal cords and articulatory problems, such as stuttering, lisping, and cleft palate, all affect the sound and clarity of speech. In other words, the actual frequency pattern of a specific uttered consonant or vowel has a large variance.

Environmental effects and the brain processing input from other modalities, such as sight or touch, can greatly affect speech. This *Lombard effect* describes an involuntary tendency to increase volume, change pitch, or adjust duration and sound of syllables in response to external noise. This compensation effect increases the auditory signal-to-noise ratio of the speaker's spoken words. This is one reason why automatic speech recognition algorithms trained in a quiet environment are difficult to transform to noisy environments. For example, an algorithm trained in a developer's cubicle will rarely work in a car.

The *spectrogram* (see Figure 2) is a standard tool for visualizing and further analyzing sound patterns. A spectrogram is an image that shows how a signal's spectral density varies with time—that is, it shows the distribution of the energies in different frequency bands in time.

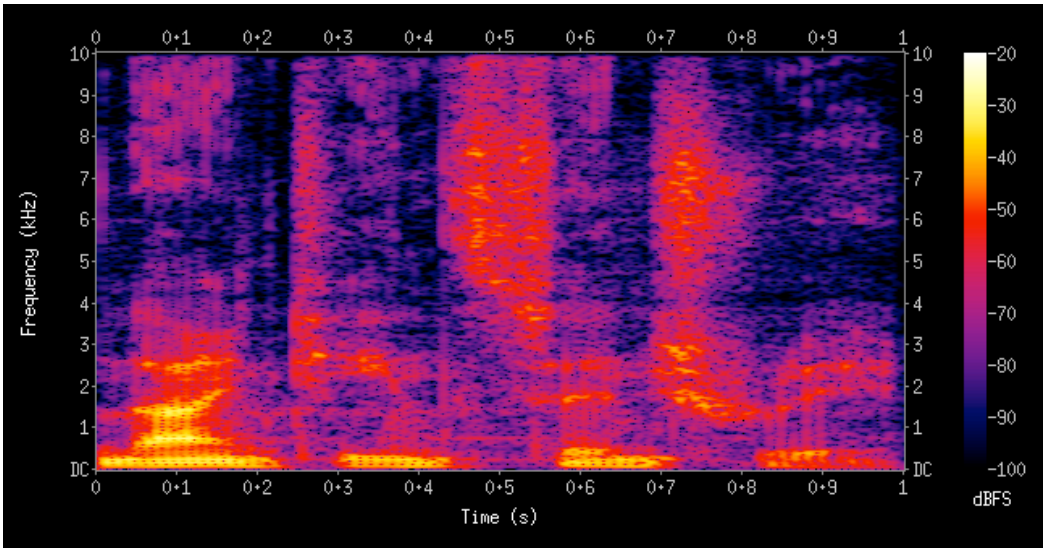


Figure 2. A spectrogram of a male voice saying: "nineteenth century." The yellow bands of energy are the formants. (Source: Wikimedia commons)

Each phoneme is distinguished by its own unique pattern in the spectrogram. For voiced phonemes, the signature involves large concentrations of energy, the *formants*. Formant values vary widely from person to person. So, reading a spectrogram mainly means recognizing patterns that are independent of particular frequencies and identify the various phonemes with a high degree of reliability.

Relatively static formants are found in the monophthong vowels and the nasals; formants that are more variable over time are found in the diphthong vowels and the approximants. The monophthong vowels, which have the strongest and most stable formants, can usually be easily

distinguished by the frequency values of the first two or three formants, which are called F1, F2, and F3. Depending on the phoneme, F1 varies from about 300 to 1,000 Hz, F2 from 850 to 2,500 Hz, and F3 from 2,300 to 3,000 Hz. Higher formants such as F4 and F5 are no longer used for communication, but are indicative of the speaker's voice. As a result of the low bandwidth and the Nyquist theorem, F4 and F5 are usually lost in telephone speech, as are many of the speakers' individual voice characteristics.

Unvoiced speech sounds are not usually said to have formants. Still, plosives are usually recognized as a great burst of energy across all frequencies occurring after a short relative silence. Aspirates and fricatives are recognized as "large clouds" of smooth energy along both the time and frequency axes.

Properties of Human Created Sounds

From a technical perspective, normal, conversational speech differs from general acoustics in the following ways:

Limited bandwidth: The spectrum and clearness (harmonicity) of the human voice varies with age, gender, and also with the language spoken. While the audible spectrum of sounds is about between 16Hz and 22kHz, speech as generated by humans usually does not go below 80Hz and does not exceed 5kHz. Differences from this general rule might exist when examining individuals. However, it is generally assumed that speech sounds below 80Hz in adult males and below 100Hz in women and children are disregarded by the human ear. The pitch of the human voice is between 120 and 160Hz for adult males and between 220 and 330 Hz for women and children. Vowels, are most important for the intelligibility of speech, they can reach frequencies up to 5kHz. The highest, frequencies are emitted by sibilants, such as "s" or "f". The frequencies can easily reach into the non-audible spectrum (above 20kHz). Consequently, to capture the whole frequency range of language a 16kHz sampling rate is currently the gold standard, which as of the Nyquest theorem, guarantees the reproducibility of 8kHz. Telephone systems usually use 8kHz, which allows the reproduction of a signal up to 4kHz. This still allows to capture most of the vowels, but consonants and sibilants are only understandable in context. This is why spelling on the phone usually has to be performed in whole words "Alpha", "Beta", "Charlie" rather than "a", "b", "c".

Limited volume: The dynamic of speech is relatively high compared to many other sounds sources, such as some musical instruments, however, in general the volume of human voice is limited to the sound energy the human body can produce. Motorbikes or gun shots can produce a much higher energy level, for example. In a 60 cm distance from the mouth, the typical sound

volume of the human voice is about 60dBA. A stronger voice can raise the volume by about 6dB. Yelling measures about 76dBA (males) and 68dBA (females). The sound volume is reduced by about 4dB every time the distance to the microphone is doubled or increased by 4dB when the distance is halved. In general, 16bits per sample are used to represent the dynamics of speech completely. However, so-called plosives like "p" or "t" can easily cause clipping, even with a well-defined capturing environment.

Limited variance in harmonicity: In contrast to generic audio, speech usually has a certain characteristics governed by the underlying language. This is similar to instruments: A violin for example almost exclusively generated harmonic sounds, while a drum almost exclusively generates inharmonic sounds. Languages usually dictate a certain ratio between vowels and consonants, which translates into a constant ratio of voiced and unvoiced sounds. So, when the language is known, this characteristics can for example be exploited in a predictive coder (see Chapter XXX). In general, the properties of the governing language will also have an impact on the expected dynamics and frequency range of the uttered speech.

It is important to remind ourselves at this point that the human vocal tract can create much more sounds than would generally be classified as speech. Apart from noise imitations (have you never tried to meouw back to a cat?), the most important one is singing. If you ever try to sing into a cellphone, you will notice that the quality isn't really as good compared to regular speech. This is because the methods used in cell phones assume normal, conversational speech, as will be explained in Chapter XXX. Also, the classification "normal" and "conversational" is important because human speech can differ significantly in other situations, such as in emotional states of anger, happiness, or enragement. Whispering is a yet different form of speech produced by humans that has very distinct properties, as has yelling. The properties of the human voice also changes with health states, such as Alzheimer disease, drug influence, or simply a stuffy nose will change different characteristics of speech. Also, as the anatomy of the vocal tract changes, so does the produced speech. Speech signals can therefore be used to not only identify speakers but also to determine the age or the height of a speaker (see references).

Production of Music

Researchers have discovered archaeological evidence of musical instruments dating as far back as 37,000 years ago. The building and use of musical instruments vary with history and culture as do the sounds that these instruments produce and the musicians playing them. For multimedia computing, we are interested in determining instruments' general properties so we can leverage

them for compressing audio, detecting instruments, and manipulating or artificially synthesizing recordings.

The *fundamental frequency*, abbreviated as f_0 or F_0 (speak: f-zero), is the inverse of a period length of a periodic signal. Pitch represents a sound's perceived fundamental frequency. Although the fundamental frequency can be precisely determined through physical measurement, it might differ from the perceived pitch because of overtones. An overtone is either a harmonic or partial (nonharmonic) resonance. In most musical instruments, the frequencies of these tones are close to the harmonics. The harmonic of a wave is a component frequency of the signal that is an integer multiple of the fundamental frequency. For example, when the fundamental frequency is f, the harmonics have frequencies f, 2f, 3f, 4f, and so on. The most important property of the harmonics is that they are all periodic at the fundamental frequency. In other words, the sum of the harmonics is also periodic at that frequency.

Timbre describes the quality of sound that distinguishes different types of sound production, such as different musical instruments. The frequency spectrum and time envelope are two physical characteristics of sound that mediate the perception of timbre.

Spectrum is the sum of the distinct frequencies emitted by an instrument playing a particular note, with the strongest frequency being the fundamental frequency. When an instrument plays a tuning note (for example A = 440 Hz), the emerging sound is a combination of frequency components, including 440 Hz, 880 Hz, 1,320 Hz, and 1,760 Hz (harmonics), and some partials. The relative amplitudes of these different spectral components is responsible for each instrument's characteristic sound.

The model typically used to describe a timbre's time envelope divides sound development into four stages: *attack* (the time from when the sound is activated to its reaching full amplitude), *decay* (the time the sound needs to drop from maximum amplitude to sustain level), *sustain* (the volume level the sound is at until the note is released), and *release* (the time needed for the sound to fade when the note ends). This is also known as the *ADSR envelope*. Psychoacoustics uses the word tone quality and tone color as synonyms for timbre.

The three main categories of musical instruments in the western world are string, wind, and percussion. A string instrument, such as a violin or a guitar, produces sound by vibrating strings. The strings' vibrations have the form of standing waves that produce a single fundamental frequency (pitch) and all harmonics of that fundamental frequency simultaneously. These frequencies depend on the string's tension, mass, and length. The harmonics make the sound timbre fuller and

richer than the fundamental alone. The particular mix of harmonics present depends on the method of excitation of the string, such as bowing or strumming, as does the timbre. The sound timbre is also significantly affected by resonances in the body of the instrument itself.

A wind instrument contains some type of resonator, usually a tube, in which a column of air is set into vibration by the musician blowing into the end of the resonator. The length of the tube determines the vibration's pitch. The length is usually varied artificially by manual modifications of the effective length of the vibrating column of air—for example, by covering or uncovering holes in the tube. The sound wave travels down the tube, reflects at one end, and comes back. It then reflects at the other end and starts over again. For a note in the flute's lowest register, for example, the round trip constitutes one cycle of the vibration. The longer the tube, the longer the time taken for the round trip, and so the lower the frequency.

A percussion instrument produces sound by being hit, shaken, rubbed, scraped, or any other action that sets it into vibration directly. The acoustics of percussion instruments is the most complex because most percussion instruments vibrate in rather complex ways. In general, at low-to-medium amplitudes, their vibrations can be conveniently described by the terms introduced in this chapter. At large amplitude, however, they might show distinctly nonlinear or chaotic behavior. Percussion instruments have the highest variance in frequency and amplitude range and are therefore the most difficult to process.

Many musical pieces contain a mixture of instruments, including human voices. Once mixed, separating the individual instruments would require an adequate model of each instrument's behavior in its environment and with the recording equipment used. For this reason, music is not only recorded and digitized but also saved in a note-like format, called MIDI, that defines a protocol to control electronic instruments. Electronic instruments have long tried to mimic traditional ones through a process called music synthesis, which we briefly describe next.

Synthesis of Sound

The artificial generation of speech and music is called *synthesis*. The first music synthesizers date back to 1876. Then, as today, the main goal was not necessarily to correctly imitate a physical musical instrument. Often, the goal was to create new sounds of artistic value. The difficulty and complexity of exact simulation of a real instrument depends of course on that instrument's properties. It's easier to simulate a flute than a piano or an organ. It's not unusual for algorithms to be invented for a particular subtype of instrument. In general though, modern music synthesis is performed by physical modeling of the instrument as well as incorporating original samples of the instrument—the *wavetables*.

Research has recently converged to apply these synthesis techniques to speech. Synthesized speech is often created by concatenating pieces of recorded speech from a database, so-called *concatenative synthesis*. Systems currently differ in the size of the stored speech units. A system that stores phones or tuples of phones (*diphones*) provides the largest range of possible synthesized output but might lack clarity and naturalness in the produced voice. Trading off this output range for usage in a specific applications, the storage of entire words or even sentences allows for higher quality output but the lowest range of possible outputs. The database is usually combined with a model of the vocal tract (such as LPC, see chapter XXX) and other human voice characteristics to create a completely synthetic voice output. This concept of *adaptive concatenative sound synthesis* is the same as for both speech and music synthesis.

Index Terms

speech,

Exercises

1. Which differences would you expect to see in a spectrogram between male and female speakers? Which would you expect to see between younger and older speakers?

2. What is the typical spectrogram of a flute, a violin, or a drum?

3. Implement an ADSR envelop filter and play around with it. Apply it to different sounds and waveforms, including noise.

Literature

A. C. Bickford, R. Floyd. *Articulatory Phonetics: Tools for Analyzing the World's Languages*, 4th edition, SIL International, July 2006.

T. Kientzle. *A Programmer's Guide to Sound*, Addison-Wesley, October 1997.

Web Links

MIDI specifications: http://www.midi.org/

Comp.Speech FAQ: http://www.speech.cs.cmu.edu/comp.speech/

Neck anatomy: http://training.seer.cancer.gov/head-neck/anatomy/overview.html

Research Papers

É. Lombard. “Le signe de l'élévation de la voix,” *Annales des Maladies de L’Oreille et du Larynx*, Vol. XXXVII, No. 2, pp. 101–119, 1911.

Slabbekoorn H, Peet M. “Birds sing at a higher pitch in urban noise.” *Nature*, 424(6946):267, 2003.

Junqua JC. “The Lombard reflex and its role on human listeners and automatic speech recognizers,” *Journal of the Acoustic Society of America*, Jan;93(1):510-24, 1993.

Scheifele PM, Andrew S, Cooper RA, Darre M, Musiek FE, Max L. “St. Lawrence River beluga Indication of a Lombard vocal response in the St. Lawrence River Beluga,” *Journal of the Acoustic Society of America*, 117(3 Pt 1):1486-92, 2005.

Chapter 6: Light and Vision

Light is one of the most basic phenomena in the universe. The first words in the Bible are, "Let there be light!" A large chunk of the human brain is dedicated to translating the light reflected off objects and onto our retinas to form an image of our surroundings. As already discussed in Chapter 2, many human innovations have evolved around people capturing and storing that image, mostly because of it's use for communication purposes: First were the Stone Age cave painters, they were followed by the painters and sculptors of the Middle Ages and the Renaissance, then came photography, film, and digital storage of movies and photographs. Most recently, a computer science discipline evolved around computer-based interpretation of images, called computer vision. More on that later on.

In this chapter, we introduce the basic properties of light and discuss how it is stored and reproduced. We discuss basic image processing and introductory computer vision techniques in later chapters.

What Is Light?

Unlike sound, which is clearly defined as a wave with a certain frequency traveling through matter (see Chapter XXX), physicists recognize that light has both wave and particle properties. It is beyond the scope of this book to discuss the nature of light in depth. We therefore define light as the portion of electromagnetic radiation that is visible to the human eye.

Visible light has a wavelength of about 400 to 780 nanometers, which corresponds to a frequency of 405 to 790 terahertz (THz). The adjacent frequencies of infrared on the lower end and ultraviolet on the higher end are still called light, even though they are not visible to the human eye. Note that infrared light is usually captured by digital cameras unless filtered out. The traveling speed of light in a vacuum is one of the fundamental constants of nature, as it is the fastest speed observable at 299,792,458 meters per second. In addition to frequency or wavelength and speed, light's primary measurable properties are intensity, propagation direction, polarization, and phase.

The phase is the fraction of a wave cycle that has elapsed relative to an arbitrary point. One can use filters to manipulate the phase to change the appearance of light.

Polarization describes the light waves' orientation. All electromagnetic waves, including light and gravitational waves, can exhibit polarization. Sound waves in a gas or liquid do not have polarization because vibration and propagation move in the same direction. If the orientation of the

electric fields produced by the light emitters are not correlated, the light is said to be *unpolarized*. However, if there is at least partial correlation between the emitters, the light is *partially polarized*. You can then describe the light in terms of the degree of polarization. One can build filters that only allow light of a certain degree and angle of polarization, an effect that is often used in 3D vision. A pair of 3D glasses will often have filters for the left and the right eye that allow different polarized light to go through for each eye so the two eyes see images with slightly different disparity. More on that later.

Light intensity is measured in three units: candela, lumen and lux. The *candela* (cd) measures luminous intensity, which is defined as power emitted by a light source in a particular direction, weighted by the luminosity function—a standardized model of the sensitivity of the human eye to different wavelengths. Figure 1 illustrates this function (see also work by **//refs?//**). The unit is derived from the light emitted by a common candle (hence its name). A typical candle emits light with a luminous intensity of roughly one candela. The physical definition is as follows: The candela is the luminous intensity, in a given direction, of a source that emits monochromatic radiation of frequency 540×10^{12} hertz and has a radiant intensity in that direction of $1/683$ watt per steradian. A 100-W incandescent lightbulb emits about 120 cd.

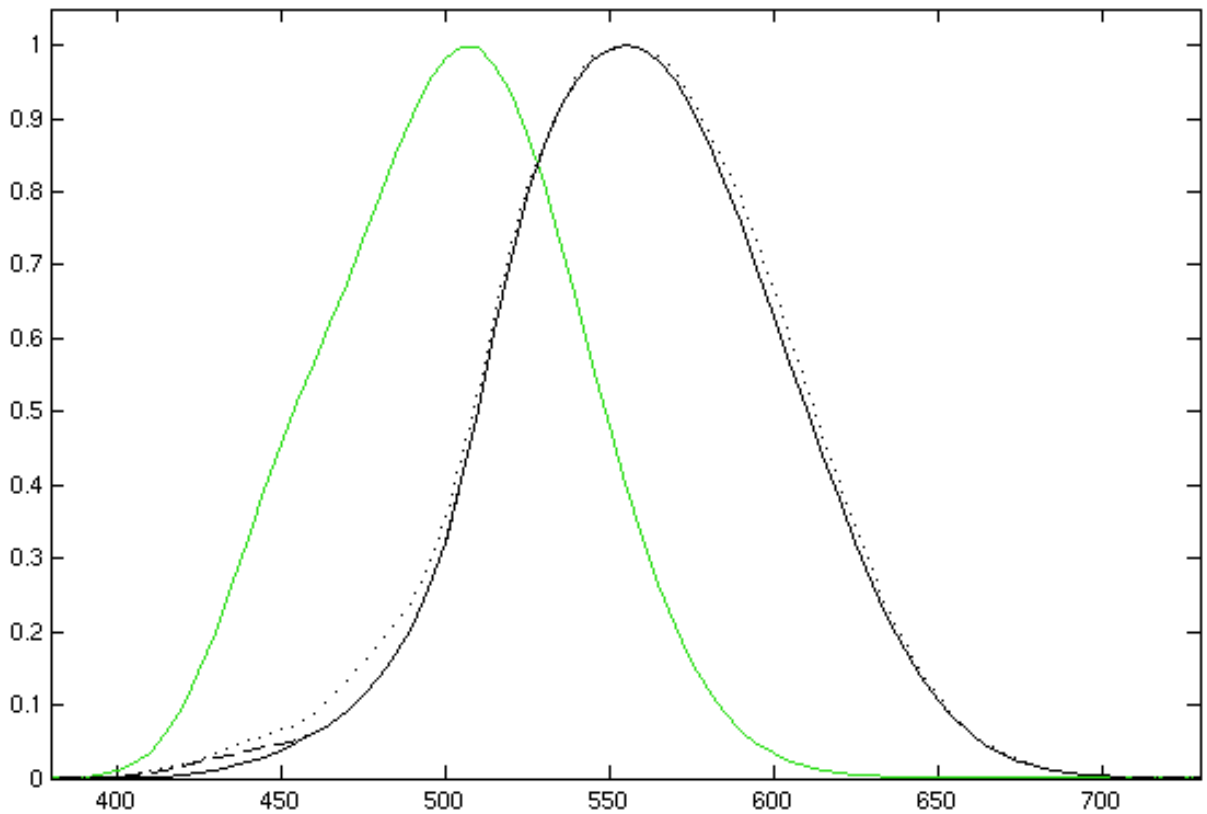


Figure 1. This graph shows different luminosity functions describing the sensitivity of the human eye to light of different wavelengths. Several luminosity functions are currently in use (see references) as measuring luminosity is still a matter of research. The dotted line is the most currently accepted luminosity function from 2005.

The lumen (lm) is the unit of luminous flux, i.e. total light emitted by an object. The lumen is defined in relation to the candela as

$$1\ lm = 1\ cd \cdot sr$$

Because a full sphere has a solid angle of $4 \cdot \pi$ steradians, a light source that uniformly radiates one candela in all directions has a total luminous flux of $1\ cd \cdot 4\pi\ sr = 4\pi \approx 12.57$ lumens. The light output of video projectors is typically measured in lumens. The America National Standards Institute (ANSI) standardized a procedure for measuring the light output of video projectors, which is why many projectors are currently sold as having a certain amount of "ANSI lumens" even though ANSI did not redefine lumen as a physical unit.

Lux (lx) is the physical unit of illuminance and luminous emittance measuring luminous power per area. The unit is equivalent to watts per m^2 (power per area) but with the power at each

wavelength weighted by the luminosity function (see Figure 1). We can convert lux, lumen, and candela into each other using the following equation:

$1\ lx = 1\ lm/m^2 = 1\ cd \cdot sr \cdot m - 2.$

A full moon overhead at tropical latitudes emits about 1 lx of light. Office lighting usually ranges from 320 to 500 lx, and TV studio lighting is at 1,000 lx. Inside the visible frequency range, humans can see as little as one photon in the dark, yet a person's eyes can be open in a desert at noon with the sun exerting up to 130,000 lx. This is an incredible adjustment that current human-made light sensors rarely can match!

Observed Properties of Light

Like sound, light exhibits properties while traveling through space. In addition, light rarely travels exclusively in a homogenous medium from a source to exhaustion. For example, lenses are typically used in recording. Moreover, the environment is full of objects that can absorb, dampen, or reflect light. Especially out-of-doors, other light sources might appear and collide with the light waves in question. Again, the resulting effects of these conditions must be considered when designing multimedia systems. For practical purposes, however, environmental conditions have a lesser impact on sound waves than on light waves.

Reflection of light is simply the bouncing of light waves from an object back toward the light's source or other directions. Energy is often absorbed from the light (and converted into heat or other forms) when the light reflects off an object, so the reflected light might have slightly different properties because it might have lost intensity, shifted in frequency, polarization, and so on. Solid objects, such as a concrete wall, usually absorb light—light waves cannot travel through these objects. On the other extreme, when light can travel through an object seemingly unchanged, the object is called *transparent*. Detecting transparent objects is probably one of the most challenging tasks in vision, including computer vision.

The most important effect observed when light passes through a transparent object is *refraction*. Refraction is the "bending" of light rays when they pass through a surface between one transparent material and another. When a beam of light crosses the boundary between two different media (including a vacuum), the light's wavelength changes, but the frequency remains constant. If the beam of light does not cross the boundary in an orthogonal angle, the change in wavelength results in a change in the beam's direction, or refraction. Refraction can be observed in everyday examples, such as when trying to grab a fish in an aquarium or observing a "bending" straw in a glass of water.

The study of light and its interaction with matter is called *optics*. Because this book is on multimedia computing, we cannot discuss light and optics exhaustively.

A common phenomena sometimes neglected but also sometimes hated by multimedia researchers is *lens distortion*. In a lens, a straight light beam hits a transparent object from varying angles, so the refraction also varies. The image that is projected through the lens is therefore also distorted. Figure 2 shows some typical distortions.

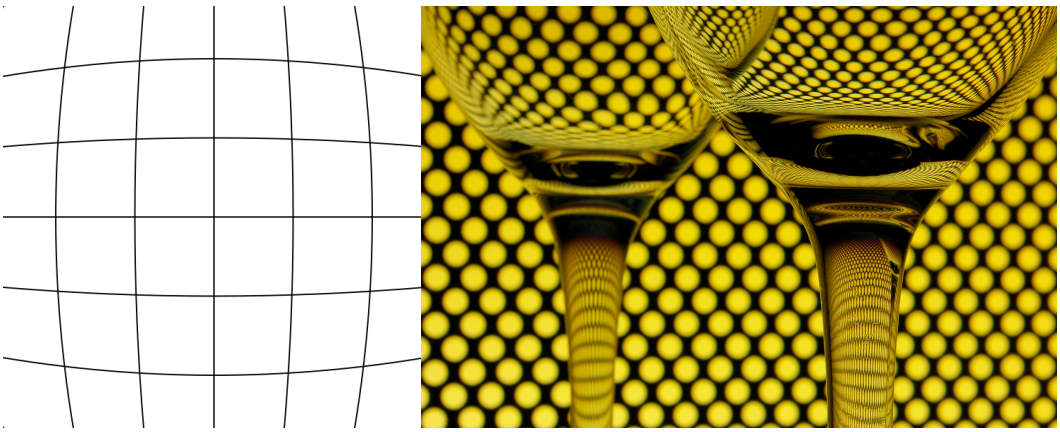


Figure 2. Left: Typical lens distortion pattern. Right: A picture from Wikimedia Commons showing the distortion created by wine glasses (picture by Atoma, creative commons license, needs to be contacted)

Correcting lens distortion can be difficult. Distortion can sometimes be corrected through calibration—that is, by projecting a well-defined object onto the lens (such as a grid, as shown in Figure 1) and calculating a correction function between the actual image and the distorted image. Often, however, you will only have the projected image, making distortion hard to correct.

Recording Light

Cameras store and reproduce light as images and video. The term "camera" comes from the Latin *camera obscura* ("dark chamber"), an early mechanism that could project light but could not store images (see Figure 3). The device consists of a box (which can be the size of a room) with a hole in one side. Light from an external scene passes through the hole and strikes a surface inside where it is reproduced, upside-down, but with both color and perspective preserved. In a modern camera, the image is projected onto a light-sensitive memory. At first this memory was light-sensitive chemical plates, later it became chemical film, and now it is photosensitive electronics that can record images in a digital format.

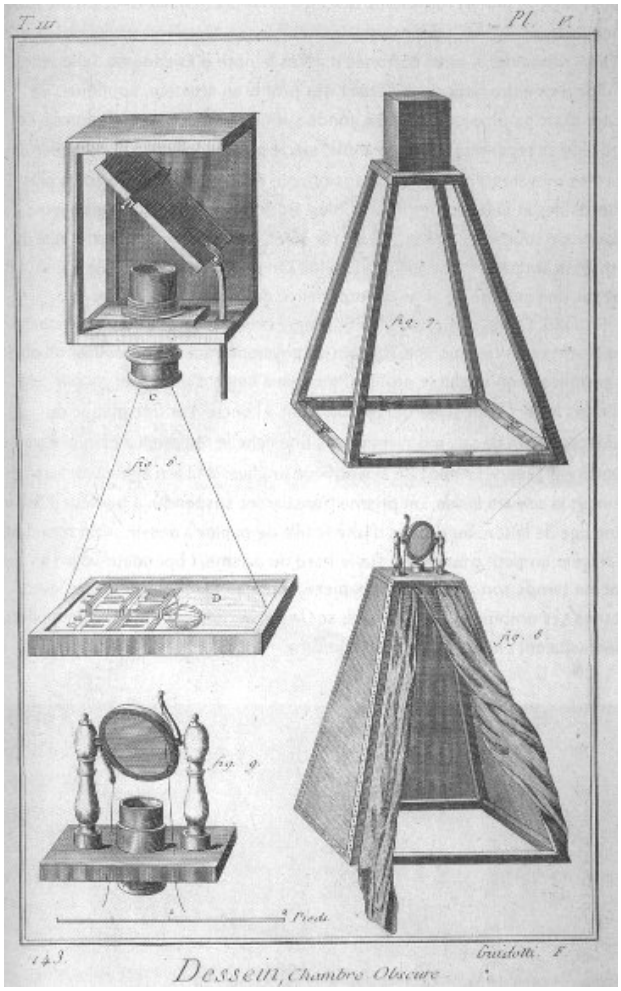


Figure 3. Historical drawings of a camera obscura often used for paintings.

Early photographic plates consisted of a glass plate covered with light-sensitive emulsions of silver salts. The salts turned light when exposed to light, leaving a gray-toned photograph behind. Photographic plates largely disappeared from the consumer market in the early 20th century, when more convenient and less fragile films were introduced. However, the professional astronomical community, which had good use for material that responds to very little light, especially in large-format frames for wide-field imaging, continued to use the plates until digital photography arrived. A chemical process converted the negative images to positive images, which are easier for the human eye to interpret—that is, light impact on the material creates a dark color stain.

Photographic film is a sheet of plastic coated with an emulsion of light-sensitive silver halide salts with variable crystal sizes that determine the film's sensitivity, contrast, and resolution. Contrary to the popular belief that you can zoom in arbitrarily into analog film because there are no pixels, analog pictures have a maximum resolution, albeit usually much higher than current

digital images. When the emulsion is sufficiently exposed to light (that is, intense enough light for a long enough time) or other forms of electromagnetic radiation such as x-rays, it forms an invisible image. Chemical processes can then be applied to the film to create a visible image. This process is called *film developing*.

Black-and-white photographic film usually has one layer of silver salts. Color film uses at least three layers. Today's films usually have many more. Dyes, absorbing to the surface of the silver salts, make the crystals sensitive to different colors. Film speed is a critical property of analog film. Despite its name, film speed is not a velocity. Rather, it describes a film's sensitivity to light exposure. The most common standardized film speed is the ISO rating. Consumer-rated films are usually labelled with ISO 100, 200, 400, or 800, where a lower number determines longer times the film must be exposed to light for a proper photograph. The speed is determined by a ratio of the optical density of the material and the logarithm of the exposure time. The logarithm is taken because film material usually reacts nonlinearly to light exposure—the reason that many professional digital cinematographic multimedia file formats still use a logarithmic sampling scale. Recording movies on chemical film usually requires taking 25 images or more per second. Of course, you must use a film with the appropriate film speed, which makes recording movies in darker light more difficult. It's also why cinematographic filming generally requires more sophisticated lighting than photography.

Today, most photos are recorded electronically. Digital cameras follow the original camera obscura principle, but instead of a chemical reaction on a plate or a film, a physical reaction occurs in an electrical photovoltaic element, typically a charge-coupled device (CCD) sensor chip. So, in other words, a modern camera is a visual sensor that converts light into an electrical signal. As with regular films, there is a maximum granularity, which in digital cameras is defined by the number of photoelectric sensors. Each sensor creates one picture element (also known as a pixel). The number of pixels is usually given as the maximum granularity of a picture, which in the digital world is called *resolution*.

Typical photo cameras have a resolution of several megapixels (millions of pixels). Resolutions of the resulting image are usually specified as X*x*Y axis resolution—for example, 1,024 × 768 or 1,280 × 1,024. Although this might change in the future, todays' digital cameras don't have the memory to store images by representing each pixel directly as a sensor value. Therefore, images are usually compressed by applying spectral compression (see Chapter XXX). JPEG format, for example, uses this type of compression. Uncompressed images (or raw images) are rare but sometimes needed for content analysis (see Chapter XXX) and for editing high-quality images.

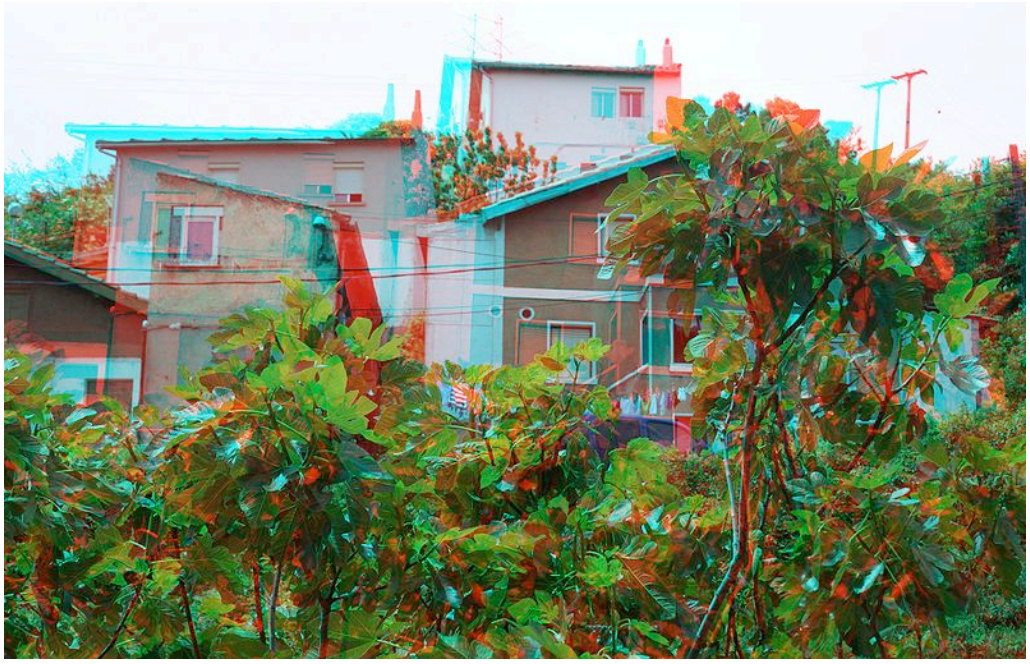


Figure 4. Anaglyph 3D photograph viewable with red/green glasses. If you are viewing this photo on a computer screen and the 3D effect does not work, adjust your display settings to match the filters in your glasses. THIS Photo must be PRINTED in COLOR!

Video cameras have recorded light electrically for a longer time. TV cameras have evolved as analog devices, storing the electrical changes on the CCD sensors on magnetic tapes and transmitting them through the air using analog radio waves. Video cameras also record sound at the same time, as we describe in Chapter XXX. For many years, cinematic cameras continued to use chemical film (usually 35-mm film) because TV cameras only delivered images with very small resolutions not suitable for the "big screen." Typical TV resolutions were PAL (Phase Alternating Line), SECAM (Sequential Color with Memory, with 720 × 576 analog picture elements), and NTSC (N*ational Television System Committee, with* 640 × 486 analog picture elements). These formats' color encodings differ, as we discuss later in this chapter.

The introduction of digital video cameras not only made the photographic and the videographic worlds converge, it also allowed videos to be recorded at much higher resolutions, especially because image compression methods could be modified to support moving pictures. Modern cameras store videos in a compressed format, such as MPEG (see Chapter XXX). The resolution of digital photo and video cameras increases constantly. As we write this book, photo cameras with up to 32 megapixels and videocameras with up to 16 megapixels are on the market.

Although photographic and cinematographic recordings can only be performed as a projection onto a surface, the resulting images do not have to be flat (that is, 2D). Of course, actual reflection in space is 3D and humans can perceive the distances between objects in space three-dimensionally. The desire to capture scenes with depth is relatively old; the first commercial 3D photo cameras date back to 1947. The stereo camera is the predominant technology for capturing 3D images. A stereo camera has two (or more) lenses and a separate photographic sensor (of film) for each length. This allows the camera to simulate human two-eyed vision, which is the basis for depth perception. The distance between the lenses in a typical stereo camera (the *intra-axial distance*) is usually the distance between a human's eyes (the *intra-ocular distance*), which is about 6.35 cm. However, a greater intercamera distance can produce pictures where the three-dimensionality is perceived as more extreme. This technique works with both images and movies, as long as images are kept separate and only one eye is exposed to each image. Therefore, for watching a 3D movie, viewers usually wear polarizing or red/cyan filter glasses (the *anaglyph technique*). These glasses separate the two images by superimposing them through two filters, one red and one cyan, or two polarizing filters. Glasses with colored/polarizing filters in each eye separate the appropriate images by canceling the filter color/polarization out and rendering the complementary color/polarization black. Although other technologies exist to create 3D projection, including autostereoscopic methods that do not require glasses, these two techniques are predominant as we write this chapter. Figure 4 shows an example of a 3D photography.

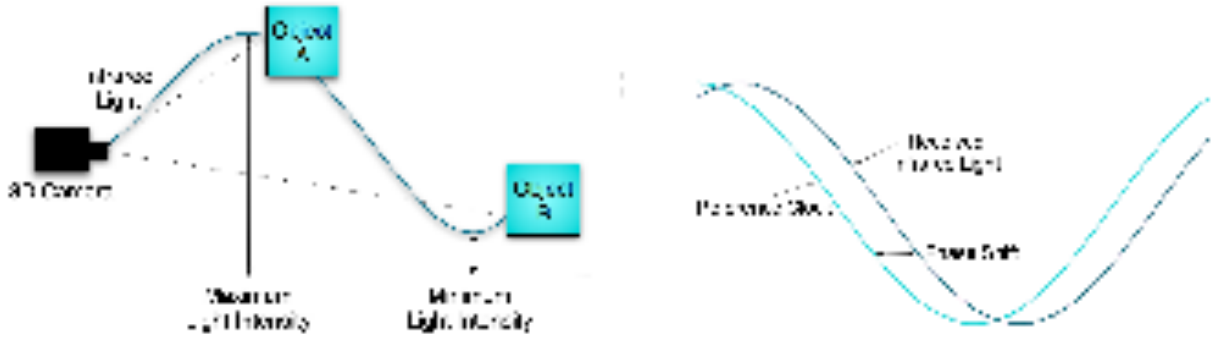


Figure 5. Left: Two objects reflect amplitude-modulated infrared light. Object A reflects more light than object B because at the point in time when the photons hit object A, they were emitted with maximum light intensity. The photons that hit object B at the same time were emitted before those that hit A, with lower intensity. Right: We can calculate the actual distance by measuring the phase shift between the emitted and the reflected light. If the distance of the reflecting object is 0, the two curves have no phase shift. The farther away the object is, the greater the phase shift.

Most importantly, these techniques aim at human perception and require the human brain to "decode" the stereoscopic image. Computing the depth encoded in a stereoscopic image is still an open research area (compare references). Therefore, different devices that try to estimate depth information in a way that it is directly available to a computer are currently under development. One of these technologies, the *time-of-flight camera*, works similarly to radar. It consists of an amplitude-modulated infrared light source and a sensor field that measures the intensity of back-scattered infrared light. The infrared source constantly emits light that varies sinusoidally.

In Figure 5, object A reflects almost the maximum intensity whereas object B, being further from the camera, reflects less light. This is because at any specific moment, different parts of the sinus wave reach the objects. When the incoming light hits an object, it is compared to the sinusoidal reference signal, which triggers the outgoing infrared light. The phase shift of the outgoing versus the incoming sinus wave is then proportional to the time of flight of the light reflected by a distant object. Thus, by measuring the incoming light's intensity, the phase-shift can be calculated and the cameras can determine the distance of a remote object that reflects infrared light. The output of the cameras consists of depth images and a conventional low- resolution gray-scale video as a byproduct. Although the idea is promising, current technological realizations still face problems with artifacts caused by quickly moving objects, light scattering, background illumination, or the measurement's nonlinearity. Last but not least, time-of-flight cameras still require a larger budget than regular cameras. Also, the reproduction of a time-of-flight recording in 3D is not straightforward.

Reproducing Light

There are two main methods for reproducing a specific light pattern:

Subtractive methods rely on intensity variations of the reflection of ambient light and do not work when no light is present. Paper, for example, reflects patterns differently once it has been modified by ink or toner.

Additive methods work with active light sources that are mixed together. The most common example is the cathode ray tube (CRT) display in a TV, as we explain next.

Photographic plates and film rely on light reflection for reproducing light, sometimes with help from a projector with a powerful light bulb—for example, for movies or transparencies shown to a larger audience. However, recording light electrically allows for an active reproduction using light sources.

As mentioned earlier, the first electronic storage and transmission of light was through TV equipment. As a result, the TV was the first technology for reproducing (moving) images electrically. Although TVs debuted in the late 1920s, television really took off in the 1940s after World War II. These TVs adopted CRT technology, which was invented by German Telefunken in 1934. A CRT is a vacuum tube with a source of electrons projected onto a fluorescent screen. The fluorescent material on the screen reflects light when hit by the electron beam. The beam is controlled by an electromagnetic field that accelerates and/or deflects the beam to control its impact on the fluorescent surface, thereby controlling the amount of reflection, and forming a grayscale image. Color TV, which was not widely available until the 1970s, uses a CRT with three phosphors, each emitting red, green, or blue light. The reflective phosphors are packed in clusters called *triads*. Roughly speaking, one triad corresponds to one color pixel. CRTs use red, green, and blue most perceivable colors can be created using different strengths of these three colors. Modern graphic cards and displays still use the RGB triad, even though the eye's perception of light uses different base frequencies.

One important characteristic of CRT triodes is the *gamma*, a nonlinear function between applied voltage of the electron gun and light intensity in the reflection. The nonlinearity of the image response often results in artifacts that are perceived as image distortions, especially in dark images. Therefore, a *gamma-correction function* is often applied to help normalize the perceived image.

Two other important CRT characteristics are its vertical and horizontal frequencies, which describe the frequency with which the beam visits each line and each column of the display. In analog TV, vertical frequencies were usually adjusted to fit the frequency of the power outlets; therefore, US NTSC uses a 60-Hz base frequency and European-based PAL and SECAM use 50 Hz. To double the perceived frame rate, analog TV uses interlacing, which allows fast motions to appear unjittered and works by only updating every second line of screen each frame—first the odd lines, then the even lines, then the odd lines again, and so on. As a side effect, the signal bandwidth can be halved because only half of the information has to be transferred per frame.

Although analog TV's advantages outweighed its disadvantages, the disadvantages become clear when compared to high-resolution digital TV. In addition, digital video compression and content analysis algorithms often perform suboptimally with interlaced video. Figure 6 shows an example of typical interlacing artifacts, the *interline Twitter*. Modern digital video encoders usually include a feature for de-interlacing image frames. Because the lost information cannot be precisely restored, however, de-interlacing algorithms use heuristics to guess the content of the lost lines. A common method is to duplicate lines or to interpolate between two lines.

Figure 6. Two interlaced video frames showing a fast motion.

Modern TVs and computer monitors do not use CRT. Instead, they use technologies that allow higher resolution in update time and image granularity, save energy, and are less bulky than CRT displays, thereby allowing larger screen size.

Plasma displays, for example, are essentially gas-filled fluorescent lamps that rely on plasma cells to display the pixels **//correct?//**. A display typically consists of millions of plasma cells compartmentalized between two panels of glass. The cells hold a mixture of noble gases and a small amount of vaporized mercury. Applying voltage to the vaporized mercury converts the gas in the cell to plasma. When the electrons flow through the cell, striking the mercury, the energy level rises, with excess energy converted to ultraviolet light. The ultraviolet light is reflected by a phosphor-painted reflective area on the back of the cell, which converts the UV light into visible light. Depending on the phosphors used, different frequency ranges of visible light can be achieved, resulting in different colors. As in a CRT display, each pixel in a plasma display is made up of a triad, so varying the voltage of the signals to the cells allows different perceived colors.

Although excellent for TV, the phosphoric reflective layer in plasmas and CRTs should not be used to display the same image for too long because doing so can damage the phosphoric layer permanently and cause image burn-in. This was the reason people invented screen savers, which ensure that the screen image changes constantly. Since about 2008, both CRT and plasma displays have given way to the even more lightweight liquid crystal displays (LCDs). Although LCDs were inferior to plasma early in their development, they now dominate the display market.

LCDs are usually more compact and lightweight, less expensive, and more reliable than CRT or plasma displays. They are available in a wider range of screen sizes, and because they do not have to use phosphor as a reflective layer (or they have no reflective layer), they cannot suffer image burn-in. LCDs are more energy efficient and offer safer disposal than CRTs.

As the name implies, LCDs are based on liquid crystals. The crystals do no emit light but can modulate light—that is, change a light wave's polarization. Each pixel of an LCD typically consists of a layer of molecules aligned between two transparent electrodes and two perpendicular polarizing filters. The electrodes are in contact with the liquid crystals and can align the crystals in a particular direction. If there is no liquid crystal between the filters, light passing through the first filter is blocked by the second perpendicular and polarizing filter and appears black. With the liquid crystals in place and without a voltage applied to the electrodes, the crystals are unaligned and modulate the light in random direction, making the display appear gray. When voltage is applied, the crystals align according to the current and modulate the light more and more in a particular direction. With enough voltage applied, the display appears black again. Varying the voltage therefore varies the amount of light passing through the filters, which is perceived as varying shades of gray. Because liquid crystals do not emit light themselves, grayscale displays have a reflective layer behind the second polarizing filter to reflect incidental light. Color displays use a light source that varies in color, usually RGB triad. The inexpensive availability of LCDs allowed TVs to become 60 inches in diagonal and larger, which also prompted demand for higher resolution, helping to popularize the HDTV standard (which was invented many years ago). Full HDTV is now at a resolution of 1,920 × 1,280 pixels and 120-Hz refresh rate.

The major challenge in reproducing 3D photos and video is creating *autostereoscopic* displays—that is, displays that do not require special viewing devices, such as glasses. Nintendo's portable game console 3DS is implementing an autostereoscopic display using the *parallax barrier* method. The parallax barrier is placed in front of the LCD. It consists of a layer of material with a series of precision-angle slits, guiding each eye to see different set of pixels based on each eye's angular direction of focus. However, because the viewer must be positioned in a well-defined spot to experience the 3D effect, the technology is used mostly for small displays, such as portable gaming systems.

Perception of Light

Multimedia computing cannot be understood without at least a basic comprehension of how human vision works. In fact, the more we learn about the mechanics of human vision, the more we

can make computer systems and algorithms adapt to it and thereby increase their (perceived) performance.

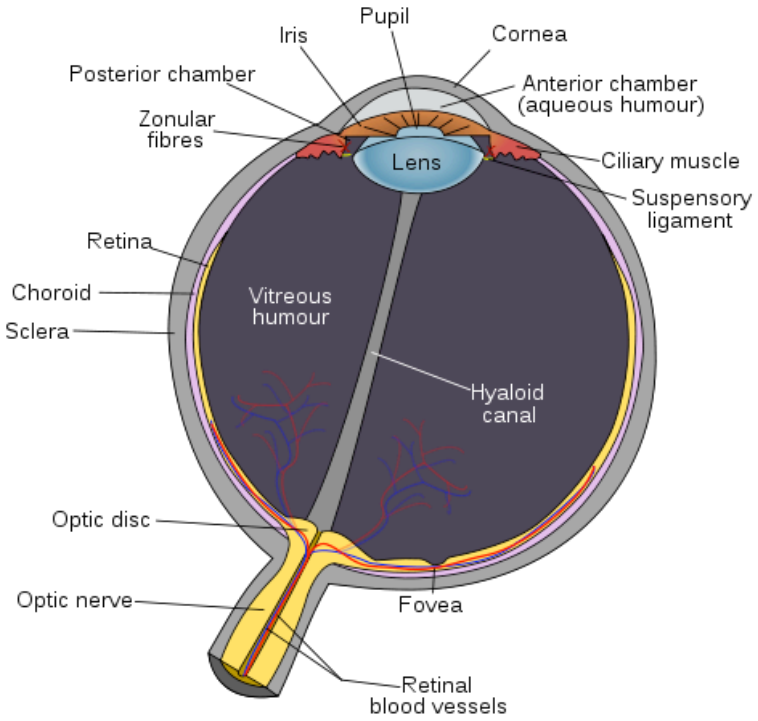


Figure 7. Schematic diagram of the human eye. The rods and cones are found on the retina.

Figure 7 shows a schematic image of a vertebrate eye. In most higher organisms, the eye is a complex optical system that collects light from the surrounding environment, regulates its intensity through a diaphragm, focuses on certain points through an adjustable assembly of lenses to form an image, converts this image into a set of electrical signals, and finally transmits these signals to the visual cortex of the brain. So, in many aspects, an eye is a complex camera obscura. We cannot explain all of the processes involved in human vision because it would fill several books and, most importantly, human vision is not completely understood. However, multimedia computing has exploited several important properties of human vision.

One of the most important properties of the human eye is that it blurs together images shown in a fast-enough sequence so they are perceived as one, enabling video. This property is present in all animals; however, different eyes have different frequency thresholds. The threshold for a human eye is about 20–25 Hz to perceive objects as a movie rather than (flickering) still images. Most

video technologies Have frame rates of 25–30 images per second (as a note: compare this to the lowest frequency acoustic stimulation is perceived as tone rather than period beats).

Like most human sensory organs, eyes perceive light intensity logarithmically (see Chapter XXX, as well as exercise X in Chapter XXX)—that is, they obey the Weber-Fechner law.

To perceive colors, the retina contains two types of light-sensitive photoreceptors: rods and cones. The rods are responsible for monochrome perception, allowing the eyes to distinguish between black and white. Rods are very sensitive, especially in low-light conditions. This is why darker scenes become increasingly colorless. The cones are responsible for color vision. They require brighter light than the rods. In humans, there are three types of cones: maximally sensitive to long-wavelength, medium-wavelength, and short-wavelength light. The color perceived is the combined effect of stimuli to these three types of cone cells. Overall there are more rods than cones, so color perception is less accurate than black and white contrast perception. This affects the variety of perceived colors in contrast to gray tones as well as the accuracy of spatial color distinction in contrast to black and white. In other words, reducing the spatial resolution of the color representation while maintaining the black and white resolution has little perceptible effect. This property of the eye has been used heavily for compression, the analog TV format NTSC for example uses less bandwidth for color transmission than for black and white transmission. JPEG image compression uses the spatial and variance color insensitivity in several ways, as we describe in Chapter XXX.

Other properties of human vision that can be leveraged in multimedia computing are based not on the eye's anatomical properties but on the brain's functional properties. These can be complex and are typically studied in optical illusions. Some, if not most, of these properties are learned. For example, if you draw a dark border on the lower right edge of a window, it will appear to be in front of the others because people have learned to interpret the dark edge as shadow. Evidence suggests that even binocular depth perception is learned (see references).

Color Spaces

This is a good time to introduce some math. As we discussed earlier, colors can be captured and reproduced by varying the intensities of fixed colors. Children learn this concept from watercolor painting in elementary school: You can use red, blue, and yellow in varying intensities to make all the other colors. CRT displays use red, green, and blue; the human eye uses yet another set of filters based on pigmenting. Mathematically speaking, we can describe all colors using a linear combination of base colors. In other words, the base colors from a 3D color space. Color spaces are an important concept because different sensors and light reproducers can only work with a

different set of fixed colors. Most printers use the CMYK (cyan, magenta, yellow, black) color space because it is most convenient for ink producers. The “K” stands for black, which could be created by mixing yellow, magenta, and cyan; however, this would be costly. So even though K is mathematically not needed, economic reasons prevail. Most importantly, color spaces are often used to analyze an image or video computationally. We present three important color spaces here and other color spaces in later chapters.

For computer scientists, the RGB color space is probably the canonical color space. Most displays, graphics cards, and raw image formats support this space. As a result, most programming tools, especially those for graphical user interfaces, work in this space by default. The RGB space is often augmented by a forth component, often called alpha, that controls the transparency of a pixel. It is important to know that the RGB color space is furthest away from human perception because contrast is not modeled explicitly. So the perceptual importance of a color component and the similarity of two colors cannot be judged easily.

For image compression, the YUV color space (and technical variants) has therefore been predominant. The YUV model defines a color space based on one luma (Y) and two chrominance (UV) components. Both the PAL and SECAM standards use the YUV color model. Previous black-and-white systems used only luma information. Color information was added separately via a subcarrier signal so that a black-and-white receiver could still receive and display a color picture transmission in the receiver's native black-and-white format. A variant of YUV is used for JPEG compression because it makes it easier to scale color and black-and-white channels independently. Conversion between RGB and YUV (and back) can be performed by a simple linear transformation:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$

The Y component is denoted with a prime symbol Y′ to indicate gamma adjustment of the Y component. A close look at the formula reveals the weighting of the different components, which corresponds to experimental evidence for human color perception. Figure 8 shows the result of this decomposition for an example image.

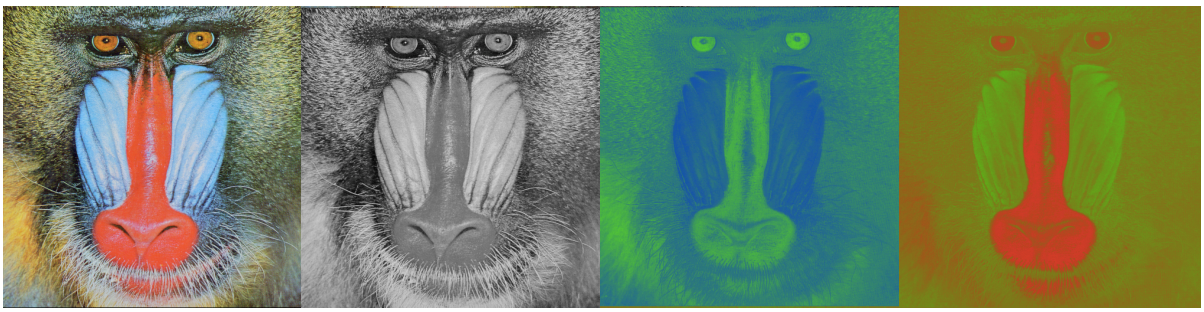


Figure 8. An image and its Y, U, and V decompositions.

As you probably suspect, a linear transformation from a color space that was invented for CRT displays cannot describe human color perception exactly enough to measure color differences. Unfortunately, although extremely important, the measurement of perceived color differences is extremely difficult because perception of color differences varies not only with lighting but also with the colors surrounding the color difference. Also, a significant number of optical illusions create "fake" colors—that is, colors that are not there but are perceived to be (see references). Obviously, the objective color difference would be zero but the perceived color difference is greater than zero. Nevertheless, one color space—CIELAB—has been created to model perceived color differences using an abundance of human-subject experiments. It has recently gained attention in the computer vision and image retrieval communities. CIELAB is designed to be perceptually uniform—ideally, the Euclidean distance between two colors reveals its perceptual difference.

The CIELAB space is based on the opponent-colors theory of color vision. The theory assumes that two colors cannot be perceived as both green and red or blue and yellow at the same time. As a result, single values can be used to describe the red/green and the yellow/blue attributes. When a color is expressed in CIELAB, L defines lightness, A denotes the red/green value, and B the yellow/blue value. Different standard illumination conditions are defined using a reference white. The most commonly used reference white is the so-called D65 reference white. CIELAB's

perceptual color metric is still not optimal, and the aforementioned assumption can lead to problems. But in practice, the Euclidean distance between two colors in this space better approximates a perceptually uniform measure for color differences than in any other color space, such as YUV or RGB. The color space uses an intermediate space, the CIE XYZ sapce. The XYZ space was designed to eliminate metamers—that is, different colors that are perceived as the same color. Figure 9 shows the color matching function used by the XYZ space.

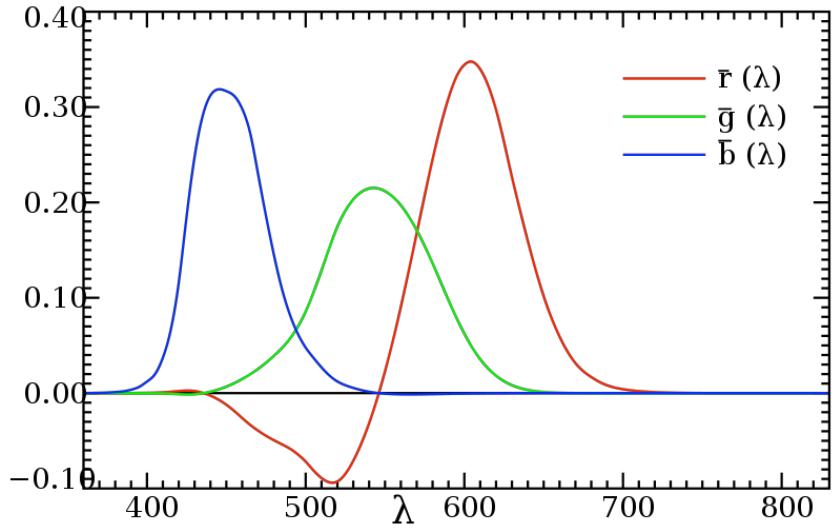


Figure 9. CIE XYZ space color matching function. The curves show the amount of primary color-mix needed to match the same monochromatic color generated by light at wavelength lambda.

The following formula converts RGB to CIE XYZ space:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Conversion from CIEXYZ to CIELAB is performed by the following formula:

$$
\begin{aligned}
L^\star &= 116 f(Y/Y_n) - 16 \\
a^\star &= 500\,[f(X/X_n) - f(Y/Y_n)] \\
b^\star &= 200\,[f(Y/Y_n) - f(Z/Z_n)]
\end{aligned}
$$

where

$$
f(t) = \begin{cases} t^{1/3} & \text{if } t > (\frac{6}{29})^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases}
$$

This chapter only gives a quick introduction to light as relevant for multimedia computing. Further properties of light, human perception, and the devices that record and reproduce light will be discussed when important in connection to concrete algorithms.

Light Production

At this point, the reader might go through the book or search the table of contents for a production of light chapter, just like the production of sound chapter, and then, however, not find it. This is not a mistake. It's remarkable that with vision being so important it seems there is no real benefit in studying the nature of light creation tools for multimedia computing yet. We sincerely hope there will be a future when holographic light sources and other light synthesizing objects are the output mainstream multimedia devices. They were already envisioned in the original Star Trek series (the "holodeck"). At the moment, light seems mainly be reproduced rather then produced. About the only thing that we could we say about light creation is that different light sources cover different spectra. The laser being the most spectrally narrow and then different light sources, such as planets or chemical reactions on earth cover different color ranges. On a different level, light creation and shaping tools include the ball pen and paper, the chalkboard, and, of course the more sophisticated computer-aided tools like vector graphics editors and 3D animation rendering. Discussing these, however, we felt belong to a different part in this book, namely part XXX (on editing).

Index Terms

light, speed of light, polarization, 3D vision, candela. lumen, lux, transparency, reflection, refraction, optics, camera, camera obscura, film developing, film sensitivity, PAL, SECAM, NTSC< interlacing, TV, intra-ocular distance, intra-axial distance, time-of-flight camera, additive light source, subtractive light source, CRT, LCD, plasma, human eye, color space, RGB, YUV, CIELAB, CIEXYZ.

Exercises

1. List the factors that contribute to an object reflecting more light than another one.
2. When a powerful light source and a not so powerful light source are placed adjacent to each other, the less powerful light source might appear to not even emit light (for example, a small LED in the midday sun). Explain.
3. Write a program that can correct lens deformations using a calibration process—that is, the photographed shape is known and a function is fitted to correct the photograph to the actual shape.
4. How much space is needed to store a raw image in NTSC and full HDTV format?
5. Take a pencil and hold it in front of your eyes. Close one eye, observe, open it again, then close the other eye and observe again. Repeat the experiment with the pencil at different distances in front of your eyes. What can you observe?
6. Describe a procedure to calibrate a 3D display with anaglyph technology and with parallax barrier technology.
7. How many bits are needed to store a pixel in CIELAB space?
8. Explain which part of visual perception is most often utilized by magicians doing magic tricks.
9. What is the equivalent of sound synthesis in the visual domain? What is the main issue when doing this?

Literature

- Wyzecki, G., and Stiles, W.S. (1982). *Color Science: Concepts and Methods, Quantitative Data and Formulae* (2nd ed.). Wiley-Interscience.
- CIE. (1932). *Commission Internationale de l'Éclairage Proceedings, 1931*. Cambridge: Cambridge University Press.
- Guild, J. (1931). The colorimetric properties of the spectrum. *Philosophical Transactions of the Royal Society of London*, A230, 149-187.
- Stiles, W.S., and Burch, J.M. (1955). Interim report to the Commission Internationale de l'Éclairage Zurich, 1955, on the National Physical Laboratory's investigation of colour-matching. Optica Acta, 2, 168-181.
- National Television System Committee (1951–1953), [Report and Reports of Panel No. 11, 11-A, 12-19, with Some supplementary references cited in the Reports, and the Petition for adoption of transmission standards for color television before the Federal Communications Commission, n.p., 1953], 17 v. illus., diagrs., tables. 28 cm. LC Control No.:54021386.
- *ITU-R BT.470-6, Conventional Television Systems*
- ITU-R Recommendation BT.709, *High-definition Television*

Web Links

- Atlas of Visual Phenomena: http://lite.bu.edu/vision/applets/lite/lite/lite.html

- Silencing: http://visionlab.harvard.edu/silencing/

Research Papers

- Crane, R.J. (1979). *The Politics of International Standards: France and the Color TV War*, Ablex Publishing Corporation.
- Fausto Bernardini, Holly E. Rushmeier (2002). "The 3D Model Acquisition Pipeline". *Comput. Graph. Forum* 21 (2): 149–172
- Brian Curless (November 2000). "From Range Scans to 3D Models". *ACM SIGGRAPH Computer Graphics* 33 (4): 38–41
- Song Zhang, S., and Peisen, H. (2006). "High-resolution, real-time 3-D shape measurement" (PDF). *Optical Engineering*: 123601. http://www.math.harvard.edu/~songzhang/papers/Realtime_OE.pdf.
- Gonzalez, F., and Perez, R., Neural mechanisms underlying stereoscopic vision, *Prog Neurobiol*, 55(3), 191-224, 1998.
- Qian, N., Binocular Disparity and the Perception of Depth, Neuron, 18, 359-368, 1997.
- Zitnick, C.L. and Kanade, T. (2000). “A Cooperative Algorithm for Stereo Matching and Occlusion Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684.
- “A Time-Of-Flight Depth Sensor—System Description, Issues and Solutions.” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Washington D.C., USA.
- Santrac, N., Friedland, G., and Rojas, R. (2006). High Resolution Segmentation with a Time-of-Flight 3D-Camera using the Exam- ple of a Lecture Scene. Technical Report B-06-09, Freie Universit ¨at Berlin, Institut fuer Informatik, Berlin, Germany.

PART III:
MULTIMEDIA APPLICATIONS AND SYSTEMS

Chapter 7: From Sensor Data to Multimedia Documents

The concept of a document has been used over centuries as a device or mechanism to communicate information. Since the technology to store and distribute information has been evolving and changing, the nature and concept of document has been evolving to take advantage of the new media technology. At one time, one considered a document to be in the form of physical embodiment such as a book and mostly contained text as the source of information. This popular notion of a book as a document has gone through changes over the last few decades and has now transformed the notion of document to be a (virtual) container of information and experiences using digital data in multiple data formats. In this modern reincarnation of document, it is not limited to one medium, but can use different media as needed to communicate the information most effectively to a recipient. This means, however, that textual and sensoric output needs to be combined in various ways to communicate different aspects and perspectives of information related to an event or an object. This part deals with the properties of applications and systems that do exactly that.

In this chapter we start the discussion using different kinds of multimedia documents. We present concepts and techniques behind many established as well as emerging systems for preparing multimedia documents. Our emphasis is in presenting concepts and how they can be applied rather than presenting details of a particular product. Creation of multimedia documents has been a very active area and there are many popular products. One can learn how to use those products from books and manuals on those specific products. In our discussion, we will not cover specific details of any product.

What is a Document?

The most commonly used document is a book. Gutenberg's invention of moveable printing press popularized the book by facilitating creation and distribution of books. Even today, when somebody talks about a document, most people think about a book. However, to the generation of people growing up with Internet and the WWW, a book will evoke a different image. They will consider a book to be collection of text and images, presented by a person to make it coherent and complete, but frozen at a particular time. Since a book was printed on a paper with substantial efforts and cost involved, it could not be easily modified or updated. Each subsequent edition was once again carefully thought about and prepared to make it complete and remain current and complete for a foreseeable future. A book was divided in multiple chapters. Each chapter addressed a particular topic again trying to be complete on that topic. In most cases a book

organization could be represented as a tree structure, as shown in Fig. 1. This tree structure was mapped into pages. One may consider pages as necessary structure imposed by physical requirements. On one hand, the text and images in a book have to be readable and hence of some minimum size. On the other hand the whole book should be such that a normal person should be able to handle it. This can be easily accomplished by designing a book as a series of attached pages so that one could flip them in a sequence in which the material in the book is presented.

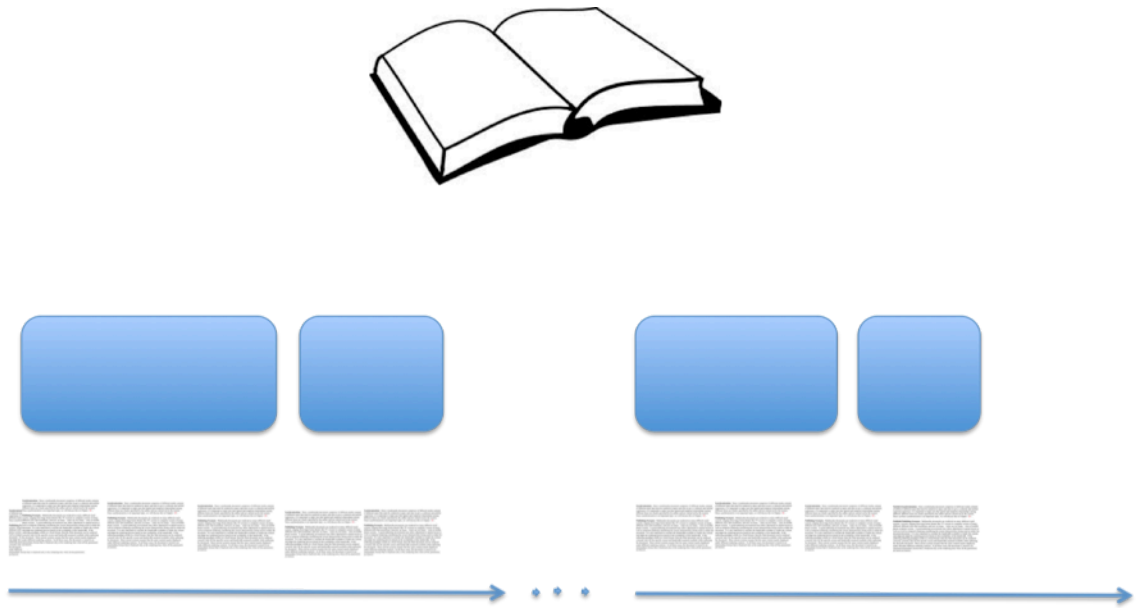


Figure 1: Organization of a book. Each page really takes an interval of the text stream and converts it into visual representation that has to be spatial – a page. Then pages are assembled into chapters, which in turn are assembled into a book.

An important thing to note in a book is that the text is now limited to pages. As we considered earlier, text is a symbolic representation of speech. And speech is a temporal signal. Thus, one may consider that a book is created by considering a timeline and dividing into appropriate intervals, which are converted into pages. Each page is a folded version of the timeline.

The concept of representation of temporal information visually on a page became dominant, if not the only, mode of creating documents due to the technology that was available. Different techniques were used to emphasize important part of the text, such as bold face, larger fonts, and different colors. Different spatial layouts were used to emphasize different aspects of text.

Arrival of audio and video technology changed the book metaphor for documents. Audio and video allow rendering of time varying audio and video signals in their natural audio and video form. The limitation of having to artificially represent time as visual pages is no longer con-

straining the types of documents that could be rendered. Now a document can contain time-varying signals in time varying form.

Another major transformation in documents is the linear nature of documents. Paper based physical representation forced linear structure on books. A book could be theoretically read in any order, but authors prepared a book assuming that usually it will be read linearly, from beginning to the end. Some people read books in somewhat random order, but most books, particularly fiction, are read in a linear order. Arrival of hyperlinks in electronic documents and then introduction of hyperlinked pages in the Web changed this notion. As we will see, now a document could be read in an order that a reader finds appropriate rather than what the author intended it to be. More importantly, now a document is no more a compact and closed physical artifact, but a dynamic organically growing artifact that could present multimedia in all its forms. And we have only seen the beginning of how future documents will be.

Evolving Nature of Documents

Most documents may be considered as a composition of many *content segments* (CS). A CS is a component that has been either authored or captured and can be considered an independent unit of media that could be combined with other segments. It is like an atomic segment that could be combined with other units to build increasingly complex documents. A CS could be a text document, a photo, a video segment, an audio segment, or any other similar data that represents a particular media. Each CS has associated meta-data that provides essential context related to its interpretation, rendering, utilization, and authorship. What is stored in meta data is dependent of the media and application domain. Some meta-data elements that have become *de facto* standard across different media are size, name, date, and place of author or device acquiring the data, and coding method to convert the media to bits. We discussed EXIF for photos earlier in Chapter <context>. One may want to look at the meta data related to text files or other data on any system to get a good feel of how meta data looks. In Figure 2, we show some content segments and elements of meta data associated with those.

File Type	Common meta data associated with the file
Text Document	Name, Author, Length, Date-Created, Date-Last-Modified,Type, …
Photo	Name, Capture-Time, Compression-scheme, EXIF, …
Audio	
Video	Name, Creation time, Compression, Length, Type, …

Figure 2: Content segments of different media and meta data commonly associated with those: a. Text document, b. Photo, c. Audio, and d. video

Almost always, the meta data about a CS is not rendered when the segment is presented, but it is always used in deciding the rendering method. Also, whenever one wants to use a particular CS, meta data is used to determine its relevance and how it could be combined for a potential new document. It is important to understand that without meta data, a segment may become unusable.

Given several relevant CS for producing a document, one may combine them in many different ways. Different combinations may result in different documents. To understand different ways to combine these, let us consider 10 different segments shown in Figure 3. In Figure 4, we show three possible combinations in which a document may use it. The first composition approach used in Figure 4a combines them in a fixed linear document that is rendered, using conventions of English text, in left to right sequence in time. In Figure 4b, these documents are composed by the author as sub-documents that could then be used as new CS that are then rendered linearly. It is possible to combine different components in many different ways. In Figure 4c, we show linking of documents so a user can go from one composite document to other if she so wishes by breaking the strict sequence that is followed in 4a and 4b.

One may consider evolution of documents along three important dimensions:

Type of Media: Until recently, most documents were mostly text documents and were commonly available in printed form on paper. Occasional photos or figures were included to enhance understanding of concepts or details that were considered too complex for text.

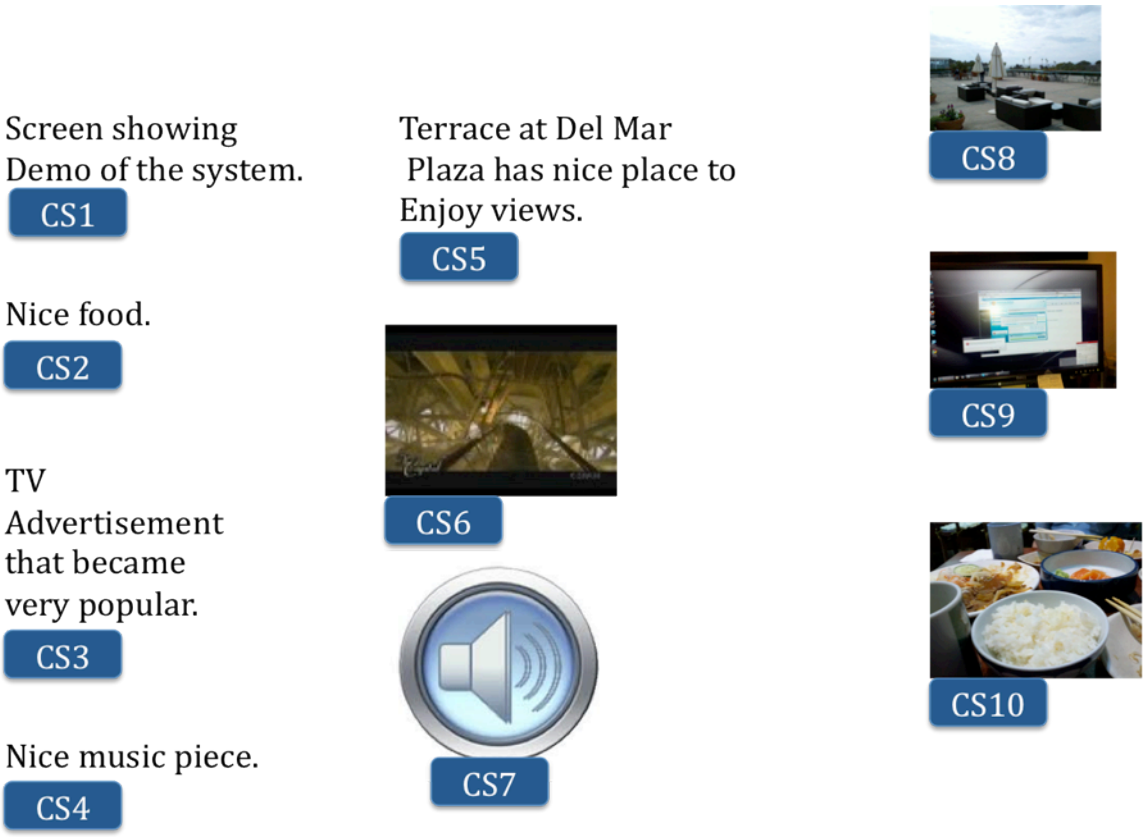


Figure 3: 10 different media segments. Each of them is independent unit and is considered a atomic unit.

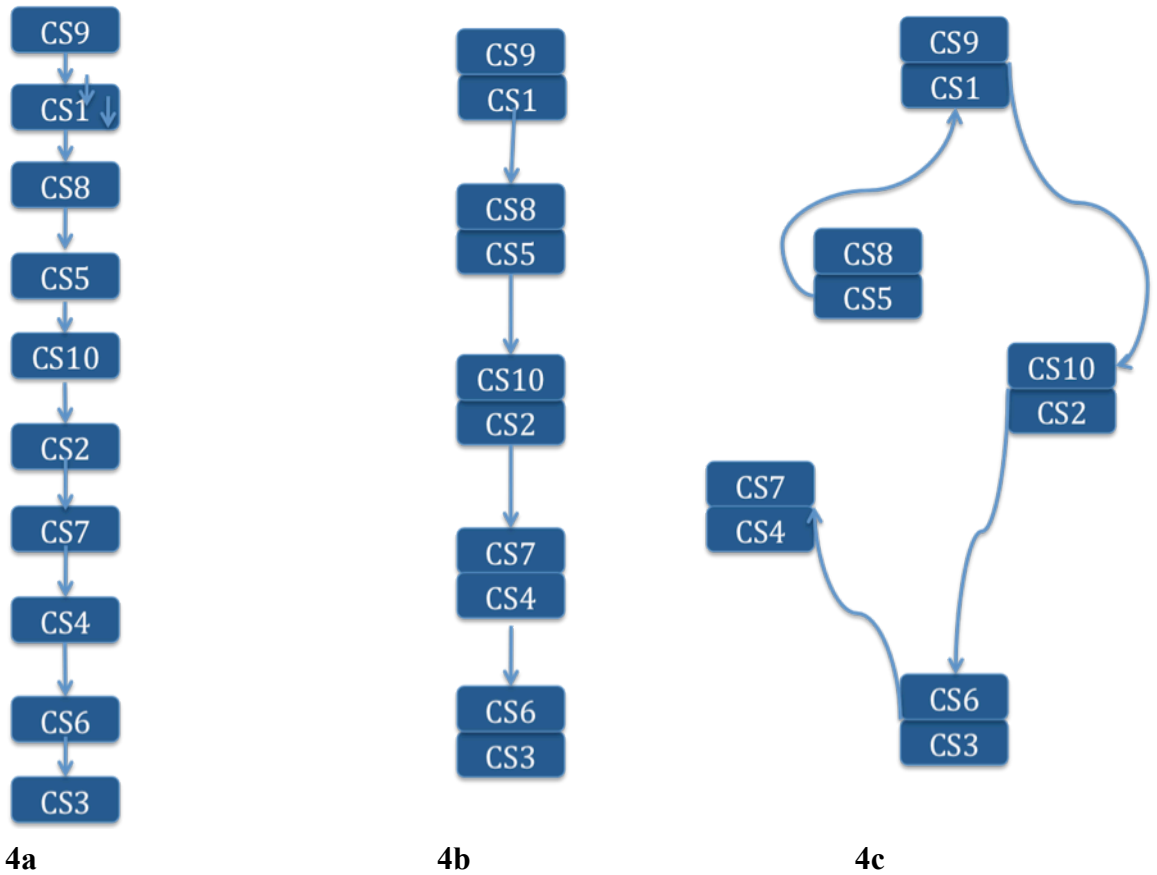


Figure 4: Three different combinations of the segments of Figure 3. In 4a, the traditional linear segment is presented; 4b shows some atomic documents combined to form composite elements and then used in the document; and 4c shows how a user may navigate from one unit to the other as she may wish.

In the last few years, due to emergence of new media technology, the nature of documents has gone through a complete metamorphosis. Text is and will remain an important component of documents, but now documents use different media as the author deems suitable for communicating the information and experiences in the best possible manner. Different media can be combined in space and time in appropriate manner by the author to communicate his ideas in the most compelling manner. Moreover, same CS could be used as many times in a document or by as many documents as need it. This is now possible because unlike in old days, a CS is in electronic form and hence could be copied and used effortlessly or linked for rendering it without copying it. This ability has resulted in revolutionary changes in creating new documents and provided new powerful approaches for expressing ideas.

Non-Linear Flow: Due to the nature of physical documents, text was designed to flow linearly. This was strongly influenced by the temporal nature of narrative structures natural to text-based

story telling. With the advent of electronic media and ability to create links, the limitation of linearity can be easily overcome. At first thought people accustomed to linear structures find nonlinearity confusing and unnatural. However, the fact that one can compose independent CS and then can link these to form multiple linear structures using different links, is making this approach very popular. As we will see in the following, this provides very flexible approach to authoring documents that may be customized for different types of audiences.

Dynamic Documents: Older documents required significant efforts to create and then were distributed using static medium such as paper. This resulted in a significant latency between an event and sharing information and experiences of the event. During early days, the latency was really large. Newspapers were invented for reducing this latency between an event and its report for important events. Television brought live events but resource limitations kept this limited to only important events. The Web brought blogs, micro-blogs, and now real time automatic updates for sharing live event information and experiences. There is an increasing trend to compile a document related to an event as it is unfolding.

In addition to such event reports being dynamically unfolding with events, they can also be designed to suit information relevance and needs of a person. This is resulting in creation of personalized dynamic doscuments.

Stages in Document Creation

Every document is created using a three step process. These three stages do have some overlaps, but are distinct enough to consider them separately.

Data acquisition and organization: When a person decides to create a document, she starts thinking about the information, experiences, and message that the document will communicate to a user of the document. This involves thinking about the relevant events and related information that must be used in the document. In some cases this information is already available to the user, while in other cases, this must be acquired. In most cases, the information available is significantly more than that could be used in the document due to the size limitations of the document. The size limitations of the document are due to attention span of the user; more than the physical requirement which in earlier times were more dominant.

A major change brought on by technology in the last few years has been the increasing availability of meta data that helps in organizing and using the data. Meta data is available for text files, as well as photos, videos, and all other data that is created or collected. In most cases, meta data

is stored with the data in the same file. This could be used for organization as well as for using the data.

Selection: An author[4] of a document usually collects lots of material in preparation for conveying the message through the document. It is common to for an author to collect significantly large volume of material in anticipation of its use in the final document. All this material must be organized so that it is available to support the author in selection of all pertinent material. Many meta-data management tools have been developed to help potential authors to organize and select such material.

The author selects the material from the content segments in the database considering the message that needs to be conveyed. The factors considered in selection of the segment are: relevance of the segment to the message, length of the segment, media of the segment, and how this segment could be combined with other segments.

This step is usually an iterative process. Once initial material is selected, the author must consider which material should be included in the final document. The author must consider the type of media available to convey the same information and experience and which one will be the best in the given context. Another important factor to be considered in the iterative process is the length of the document as well as the effectiveness of the information and media used to convey that. The output of the process is a set of segments to be included in the final document.

Editing: Editing is the process of taking an existing document and modifying it for its use in a given context or simply for refining it. Editing is media dependent. Many powerful tools have been developed for editing documents of specific media type. Here we briefly discuss some common operations used in commonly used professional tools.

Text Editing: Many tools have been developed for editing text documents. Commonly used operations involved in editing a text document are

- Insert,
- Delete,
- Format to change the layout, and
- Emphasize using different styles, sizes, and colors.

[4] We will use the term 'author' for the person who prepares a document. In some cases, like for video, usually the term 'producer' is more common. But we will use author for all kind of documents.

The first two operations are obvious for changing the text. Formatting is used to provide structure to the text and includes breaking the text into sections or paragraphs, adding footnotes or references, creating special textboxes, and similar things to provide clear visual separation on a page. The final operation of emphasizing is to clearly display relative importance of certain text segments by using bold, italics, underline, larger font, different font styles, or different colors. Human beings use different intonations and inflexions in oral communication. Since text is a static representation of oral communication, these emphasis tools are used to capture some of the characteristics of oral communication.

Photo Editing: A photo is a flat static representation of visual information. Most photos are captured using a photographic device, but there are other mechanisms such as computer graphics or human painting or sketching used for creation of photos. Some of the common operations used in photoediting are:

- Selection of important objects,
- Addition of Objects,
- Deletion of Objects,
- Enhancement or restoration of visual characteristics, and
- Changing visual characteristics in parts of a photo.

In photographs, the most important aspect is to clearly mark pixels in an image that may represent a particular object. This operation is significantly more difficult than it appears. Many tools such as magic wand and cropping are provided to facilitate this operation. Enhancement and restoration are aspects that are normally used to compensate for some artifacts introduced due to imperfections during the photo capture operation. Visual characteristics are changed in parts of photo to make visual appearance more appealing. Finally, addition and deletion of objects are fundamentally to change the content of a photo. A photo represents state of the real world captured at a particular time. By inserting or deleting an object, an editor is changing the state of the world as depicted by the photo[5].

Audio Editing: While many older audio editing tools try to simulate a tape recorder, modern editing operations on audio are usually based on a visual representation of the amplitude space (going from left to right in time). Audio can be

- cut out,

[5] Before photo editing tools became common, some editing was done in dark room. Before digital tools arrived, a photo was considered a strong evidence of what the state of the real world was at the time photo was taken. Digital photo editing tools allowed manipulation of photos and eliminated photos as an evidence of the real world.

- copied,
- pasted, and
- filtered.

The problem with most visual representations of audio recordings is that they are not intuitive, e.g. the user must listen in very often as the amplitude space representation does not indicate the final acoustic experience good enough. Several tracks are usually visualized above each other. Speech editors therefore often show a spectrogram (see Chapter XXX) of the speech signal, allowing a more intuitive representation for experts. Midi editors allow the editing of notes, which makes it easier for musicians. They work with Midi editors like a text processing tool.

Video Editing: Video is different from the above media in that it combines all of the above and adds some new dimensions. It has spatial dimension and characteristics of photos, but represents rapidly varying sequence of photos thus bringing in temporal dimensions. It is also combination of not only a photo sequence but also of audio that is either captured with the video or is added to the photo sequence. Moreover, one could either overlay text in some parts of video or even use text exclusively as video segments. Video editing tools usually contain:

- Photo editing tools
- Adding a video segment at a particular time say Ti
- Deleting a video segment from time T1 to T2.
- Add a text box at specified location from time Ts to Te
- Add an audio channel from time Ti
- Add an imagebox a specified location from time Ts to Te
- Add another video, usually of a much smaller size, at a specified location from time Ts to Te
- Add specific transition between segment Sk and Sk+1.

As can be easily seen, video editing tools utilize results of editing of all other media and must provide spatial and temporal composition operations to combine different media to provide a coherent media. In Figure 5 we show a video authoring/editing environment that contains photos, video, and audio components that are combined using timelines.

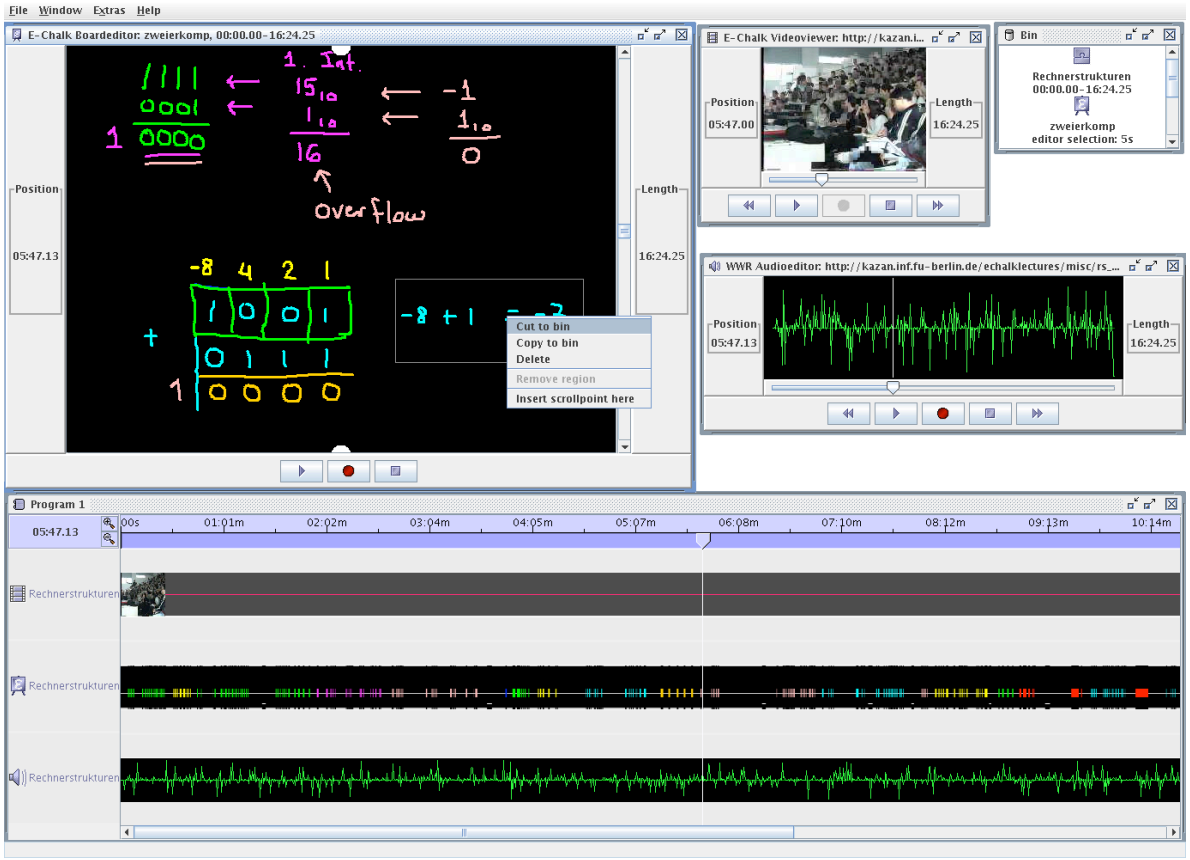


Figure 5: A video authoring/editing environment uses a timeline for showing how different components can be organized.

Emerging Multimedia Editing Tools: In a way multimedia tools are extension and collection of individual medium editing tools. Since in most of the current multimedia systems also a screen is used for rendering, spatial layout is manipulated similar to text and photos. Audio and video bring in temporal elements. This means that tools must be provided to manage time. Multimedia editing tools are similar to video editing. The major difference in multimedia and video editing tools, however, is that video editing tools consider the photo sequence as the driving medium and other media play a supporting role. In multimedia, video is considered at the same level as any other media. In fact, a good multimedia authoring environment considers that all media are equally important and one must use a specific medium to convey or emphasize important information or experience that is most relevant for communication. A multimedia authoring environment must consider elements discussed in the following section.

Basic Elements of a Multimedia Authoring Environment

Since a multimedia document utilizes all media to make a document that combines appropriate medium to communicate the message in the most compelling manner, it must provide facilities to author each individual medium and to combine them effectively and efficiently. Moreover, to facilitate interactivity of the user with the document, an authoring environment must also consider mechanisms for user interactivity at the time of authoring. Based on the emerging changes in the nature of documents, one must consider different factors in designing multimedia authoring environments. Two very important fundamental aspects are related to spatial and temporal composition of different media assets. As we see below, one must pay careful attention to layout as well as synchronization issues.

Characteristics of Media Assets: Different media assets have different spatial, temporal, and other informational attributes that play important role in the combined documents in terms of designing their layout and synchronization. In many cases these characteristics are stored as meta-data along with the data corresponding to the media. In some dynamically created content, this meta-data as well as the data becomes available only at the rendering time. An authoring environment should account for this.

Spatial Layout: Most multimedia documents in current systems are rendered on a screen. The screen has fixed spatial dimensions, such as 640 X 480 pixels or 1920 X 1200 pixels. An author decides which media item should be displayed in which area of the screen and what should be its resolution. In some cases an item must be scaled up or down to fit the selected window size. Thus, with each media item, there is an associated spatial window where it should appear. In Figure 6, we show the screen and multiple windows. Each window size must be specified using either a rectangle in absolute locations or in terms of a corner and its size. For each window, one must also specify the type of content and the source from where the content must be displayed.

Figure 6: A spatial layout showing three different windows. Each windows location and size must be specified. Windows A1 and A2 are within A; and C1 and C2 are within C.

Temporal Layout: Since multimedia content can be displayed as video on the screen, an authoring environment should specify different content that will be used to constitute this video. The earliest authoring tools in this area started appearing in video editing systems. Multimedia authoring systems extend them to include more sources of data and provide more flexibility and control in using and combining the data. An example of temporal layout is shown in Figure – <temp-layout>. This layout essentially provides tools to specify the time interval for the appearance, transitions, and disappearance of each content item on the screen. The location of the content on the screen could also be specified.

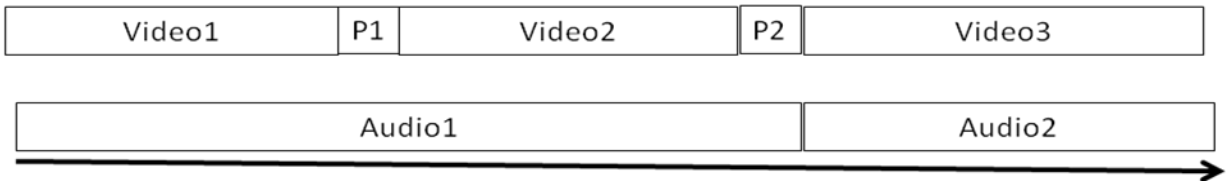


Figure 7: The timeline representation of each content item is shown on the timeline. For each item the start time and the duration must be specified.

Synchronization: A multimedia document comprises of different media contents of different types that must be rendered in space and time to give a coherent and unified experience. It is important to make sure that spatial and temporal relationships among different items are clearly specified by the author and are carried out by the system. Since synchronization is an important topic, we will discuss this in Chapter XXX. Most multimedia authoring environments provide basic tools to specify which media elements should be synchronized.

Publishing Formats: Multimedia documents are rendered on many different sized screens, ranging from large home theater like TV screens to computer screens of many different sizes and resolutions, and now on many – some say too many – sizes of mobile phone screens. A good authoring environment may allow adjustment in spatial layout as well as temporal rendering considering the screen characteristics being used to render the document. It is also important to consider the bandwidth available to render the content and adapt the rendering process based on the availability of the bandwidth. If the authoring paradigm results in a fixed format, then the final document can be rendered correctly only for the specific screen and bandwidth assumed available while authoring the document. Most current systems assume that the same content maybe displayed under different rendering contexts. In all these cases, the document is stored in an intermediate format that is finalized only at the rendering time when all the parameters are known.

Representation of a Multimedia Document

As may be obvious from the above discussion, the structure of a multimedia document is relatively complex. A text only document has fairly linear structure comprising of chapters, sections, and subsections. With modern hyper-linking capabilities, nonlinearity has been introduced in otherwise linear text documents. Now a user may play a role in defining the rendering of these documents also, as discussed earlier in this chapter. Due to flexibility in organizing spatially and temporally and use of multiple types of media, the nature of the multimedia documents becomes relatively more complex to understand. Many different models have been used to represent multimedia documents. In this section, we discuss two of these models that cover many requirements of multimedia authoring environments and have gained popularity.

Structure-based Representation: Common structure-based representation uses a tree structure in which the root is a complete document and the leaf nodes are individual media elements or a pointer to those. Intermediate nodes are 'sub-documents' comprising of combinations of indi-

vidual media elements. For each intermediate node, the composition rules and spatial and temporal layouts maybe explicitly specified. Figure 8 shows structure of a multimedia document that contains multiple text, photos, audio, and video segments. One content element could be used multiple times if desired.

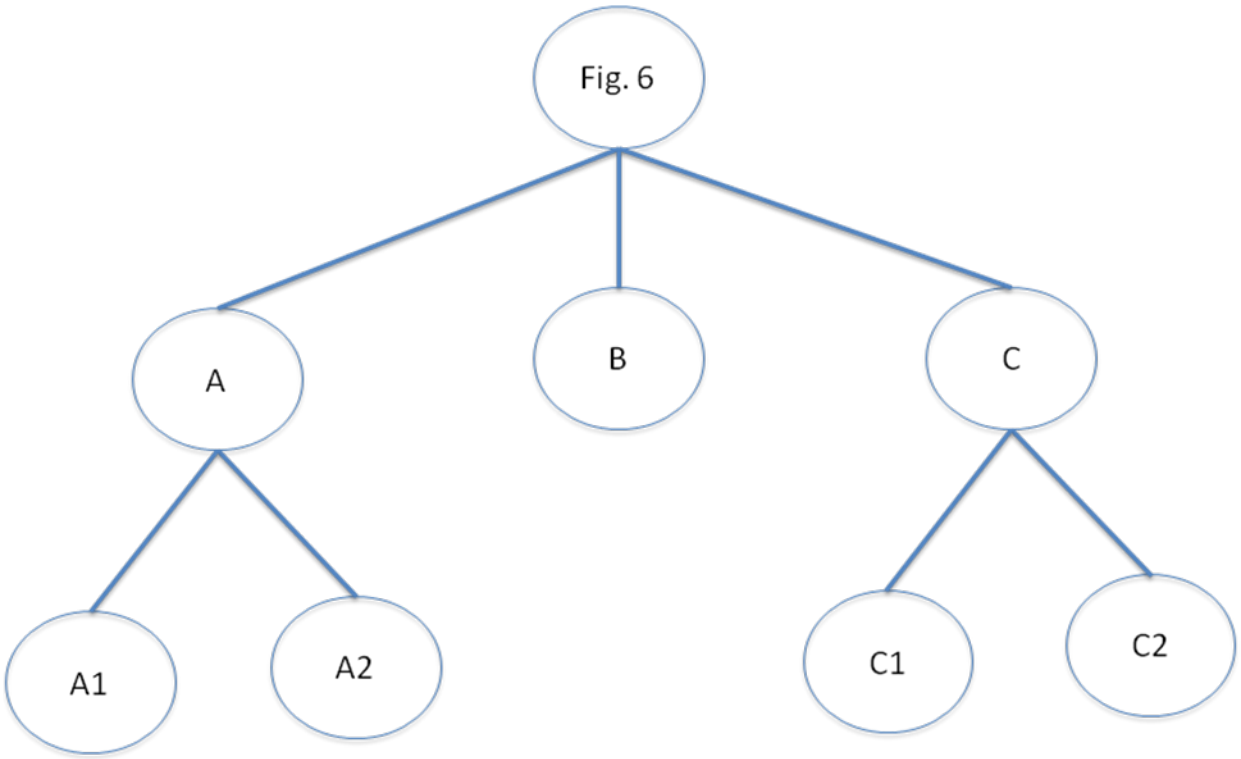


Figure 8: The tree structure shown represents a complete multimedia document that uses several media components. This structure represents the window shown in Figure 6.

Time-Based Representation: Time based representations evolved from video editing. In these representations one considers that a multimedia document is organized around a timeline. Different media elements are represented as different tracks synchronized with the master timeline. Each track specifies which content element will appear during which time interval. One may also specify the relative spatial position of each media element. This representation makes the relative appearance and disappearance of different media elements explicit and easy to represent and understand.

Current Authoring Environments

Many multimedia authoring environments have been developed in the last two decades. One of the biggest drivers for developing these environments has been the Web. Many other systems

were also defined for general multimedia authoring environments, such as MPEG4 and SMIL. Rapid convergence is taking place in devices, communication, and computing. It appears that the Web environment may become the unifying environment. In the following we briefly discuss some key concepts and trends among emerging authoring environments.

HyperText Markup Language (HTML) was defined to be the first publishing language of the Web and has remained the main language for preparing documents so they can be published on different platforms. Like any other markup language, HTML uses tags to specify how an element on a page should be published. The language syntax defines how to specify tags for different actions to be performed. These tags are in pairs like <T1> and </T1>, where the first tag declares the beginning of T1 and the second tag is the *end tag* closes it. Most of the information in text, tables, and images is between the tags. The tags are used by a browser to interpret the intent of the author in displaying the content of the page or the document. In the early days of the Web, most of the documents usually contained only text. Tags in those days usually specified presentation related operations on text. HTML1.0 was a key component of the launch of the Web and was predominantly concerned with presentation of the text on a page.

As the nature of documents changed to more multimedia, subsequent versions of HTML provided specifications for inclusion of multimedia content. These specifications had richer tags for layout of media items. Another challenge faced by browsers in the presentation of multimedia content was use of proprietary technology for playing video. The latest version of HTML, HTML5, has introduced specific features to author multimedia content as easy as text. In particular, it now has four specific constructs: <video>, <audio>, Canavas, and SVG. These features make inclusion of multimedia content in a document much easier than earlier.

Further Reading

An excellent history of development of diffeent media and the impact on society is presented in 'Cognitive Surplus' by Clay Shirky. James Gleick's book, The Information: a History, a Theory, a Flood, is an excellent source for the changing nature of information and how it has affected our society.

Photoshop was a major force in converting photos from a visual record to an authoring environment. Photos, often called images, used to be a record that could be processed to enhance them and to recover some information. By providing simple tools to edit them, Thomas Knoll's system, developed when he was a doctoral student supervised by Ramesh Jain at the University of Michigan, changed the way photos were viewed. From a a record it became a creative environment for expressing visual thoughts. The impact of Photoshop on multimedia authoring is not

only in photos, but also in video production. On the lighter side, Photoshop destroyed what used to be considered an irrefutable evidence – a photo of an event – and has now resulting in creation of multimedia forensics as a field.

An important multimedia authoring project that contributed many important ideas and resulted in development of a complete multimedia environment was Synchronized Multimedia Integration Language (SMIL). This environment developed over several years and made available to community was one of the first authoring environment to consider all aspects of multimedia authoring and make it compatible to emerging concepts and tools from the Web community.

MPEG4 was the first effort to consider video as composition of objects and events both for compression as well as for providing interactive environment dynamic visual environments. Efforts started in creating multiple perspective and immersive video in the 20th century, but due to technology limitations remained in the conceptual stages. With advances in technology, it is expected that these techniques will advance rapidly and will result in powerful immersive telepresence systems.

Finally, one is seeing emrgence of new media as a new communication mechanism for knwledge. Emerging social media systems rely on combination of multiple media to communicate and share experiences, unlike the dominant medium of text that started with Gutenberg's moveable printing press and has remained dominant so far.

Index Terms

Literature

Research Papers

'Cognitive Surplus' by Clay Shirky
The Vanishing Line Between Books And Internet
Hugh McGuire http://bit.ly/crDZok
James Gleick, The Information: A History, a Theory, a Flood, Pantheon books, 2010.

Web Links

Exercises

Chapter 8: Multimodal Integration and Synchronization

So in order to convey the structure of a document, a multimedia system uses data and information from multiple sensors to achieve its goals. Let's continue with a more formal assumption that $S_1, \ldots S_n$ are data streams from n different sources of K types of data in the form of image sequence, audio stream, photos, motion detector, and other types including text. Each data stream has $M_1, \ldots, M_n$, as its metadata that may include location and type of the sensor, viewpoint, angles, camera calibration parameters or other similar parameters relevant to the data stream. We will use this formal assumption throughout the remainder of the book.

As discussed earlier, a fundamental difference in multimedia from single medium understanding is that partial information from multiple media is correlated and combined to get complete information. Without correlating the information from multiple data streams, one can not extract information about the real world. Even in those systems, where multimedia is for direct consumption by humans, all correlated information must be presented for humans to extract information that they need from the multimedia data

The human brain can integrate different sensory modalities, such as sight, sound, and touch, into a coherent and unified perceptual experience. Experiments show that by considering input from multiple sensors, perceptual problems can be solved more robustly and even faster by humans.. This *multimodal integration*, or *multisensory integration*, is not yet completely understood, but it is fundamental to the success of multimedia systems. Multimedia computing strives to imitate the properties of multimodal integration regardless of the incomplete understanding of the mechanisms in the brain. or example, multimedia content analysis (as described in Chapter XXX) combines audio and video in an attempt to gain accuracy, robustness, and sometimes speed.

Multimodal Integration

Here, we describe some well-known observable phenomena that might help to both clarify the process and highlight the design considerations for multimedia systems.

Experiments have indicated that two converging sensory stimuli can produce a perception that differs not only in magnitude from the sum of the two individual stimuli but also in quality. The classic study, which introduced the *McGurk effect*, dubbed a person's acoustic phoneme production with a video of that person speaking a different phoneme. The result was that the user perceived neither the visual nor the acoustic pronunciation but instead heard a third phoneme.

McGurk and MacDonald explained in their 1976 article that phonemes such as ba, da, ka, ta, ga, and pa can be divided into two groups:

phonemes that can be visually confused (da, ga, ka, ta), and

phonemes that can be acoustically confused (ba and pa).

The combination of the visual and acoustically confused phonemes results in the perception of a different phoneme. For example, when an uttering of ba is dubbed on a video showing the uttering of ga, which is processed visually through lip reading, the visual modality sees ga or da, and the auditory modality hears ba or da, which combine to form the perception of da.

Ventriloquism is another important effect. It describes the situation in which acoustic tracking of a sound's origin shifts toward the visual modality. In conditions where the visual cue is unambiguous, the perception of the visual location overrides the acoustic location. Artists throughout the world use this effect. Ventriloquists manipulate how they produce sound so it appears that the voice is coming from elsewhere, usually a puppet. This is used in many multimedia systems, e.g. in ordinary Television sets, as the loudspeakers are usually not located exactly where the actors move their mouthes on the visual screens...

An almost "magic" effect is called *body transfer illusion*. Botvinick and Cohen performed the original, so-called rubber hand experiment in 1998. Human participants sat in front of a screen showing a dummy hand being stroked with a brush while they felt a series of synchronized and identical brushstrokes applied to their own hands, hidden from their view. The result was that if the dummy hand was similar to the participant's hand in appearance, position, and location, the human subject was likely to feel that the touches on his or her hand came from the dummy hand . Furthermore, several participants reported that they felt the dummy hand to be their own hand. Virtual reality applications start to try to exploit this effect and try to induce the perception of owning and controlling someone else's body (usually an avatar) by applying visual, haptic, and sometimes proprioceptual stimulation synchronously (sensors are described in more detail in Chapter XXX).

The brain exploits multimodal integration in different ways. The two most important are probably the decrease of sensory uncertainty and the decrease of reaction time. Experiments have shown that uncertainty in sensory domains leads to an increased dependence of multisensory integration. If a person sees something moving in a tree and isn't sure whether it is a bird or a squirrel, the natural reaction is to listen. If the object emits a chirp and the brain localizes the

sound to be coming from the tree, the person takes this as proof that the creature is a bird. Hence, the lack of visual information is augmented by acoustic information.

The Hershenson experiments (see references) also showed that responses to multiple simultaneous sensory stimuli can be performed faster than responses to the same stimuli presented in isolation. Participants were presented a light and tone simultaneously and separately, and were asked to respond as rapidly as possible by pressing a button. Reaction time differed with varying levels of synchrony between the tone and the light. This result is, however, hard to generalize as multiple synchronous stimuli might also cause the opposite effect, as we discuss in the next section.

Split Attention

Split attention, the opposite effect of multimodal integration, manifests when the same media (for example, visual and visual) is used for different types of information at the same time. This is usually a problem in multimedia systems, rather than a functionality.

To understand and use the materials provided, one must split attention between them. Split attention should not be confused with distraction, although the two problems are related. Distraction is caused by the lack of ability to pay attention to a particular object due to lack of interest in the object or the greater intensity, novelty, or attractiveness of something other than the object of attention. However, split attention is caused by the lack of integration of the object to be paid attention to.

Figure 2 shows an example multimedia system known to have caused a split attention problem. The E-Chalk lecture recording system showed dynamic board content in addition to a video of the lecturer. In a typical E-Chalk lecture, two areas of the screen compete for the viewer's attention: the video window showing the instructor and the board or slides window. Several researchers tracked students' eye movements as they watched a lecture recording containing slides and an instructor video in a setup similar to that shown in Figure 2 (see references). Measurements showed that a student often spends about 70 percent of the time watching the instructor video and only about 20 percent of the time watching the slides. For the remaining 10 percent of time, the eye focus was lost to activities unrelated to lecture content. When the lecture replay consists of only slides and audio, students spend about 60 percent of the time looking at the slides because they have no other place to focus their attention in the lecture recording. The remaining 40 percent, however, was lost due to distraction.

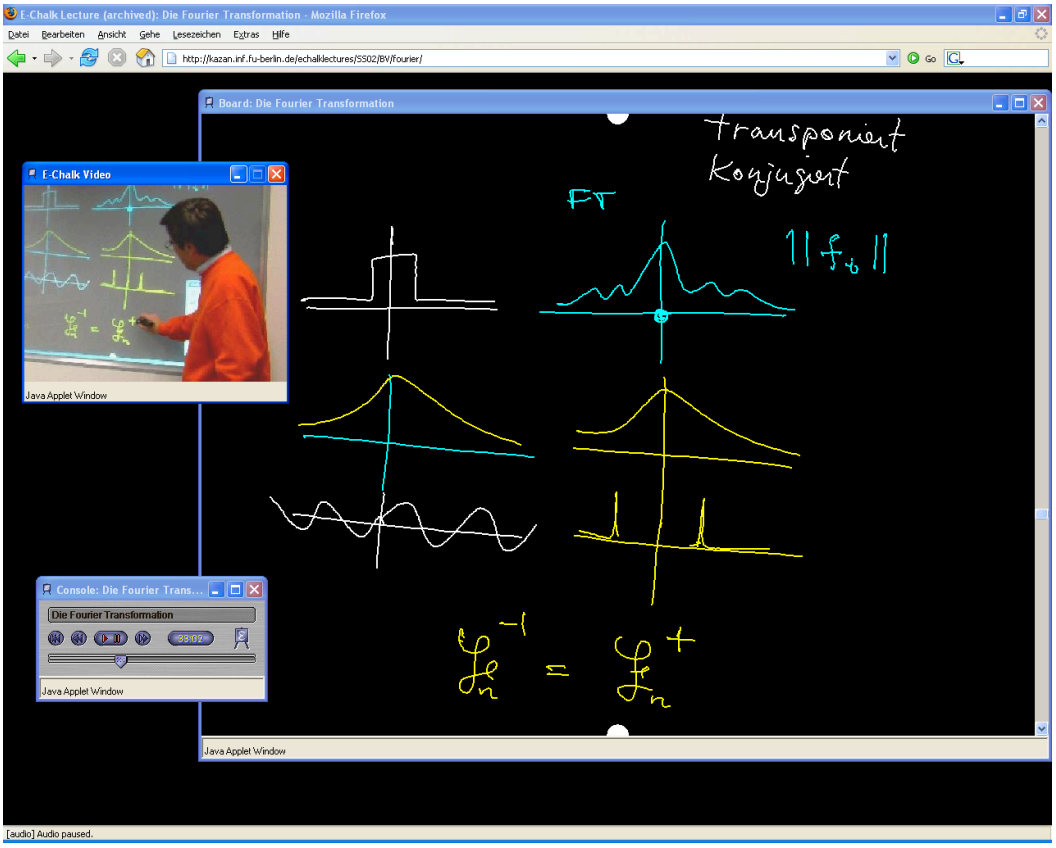


Figure 2. An example of a split-attention problem. The lecturer on the left is shown as a video, and the dynamic board content he creates is shown on the right. The additional video window is used to convey gestures and finger pointing. However, presenting the lecturer in a second window causes cognitive overhead usually referred to as split attention because the viewer switches between the content on the board and the lecturer on the left.

It is still an open question whether attention can be split to attend to two or more sources of information simultaneously. Psychologists and neuroscientists have discussed this topic for decades. Most researchers, however, now accept that attention can be split at the cost of cognitive overhead. It is this cognitive overhead that is to be avoided when designing multimedia systems. See the references at the end of this chapter for more information

So far, we have described sensory phenomena on a high level and discussed important properties of the human system. We will now begin to dig deeper into what it means to process sensory information using computers. While reading the next chapters you might find it useful to go back to these introductory chapters from time to time to remind yourself of the fundamentals. However, you will do so at the cost of some cognitive overhead, a phenomenon called split attention.

Sensor Integration in Multimedia Computing

Combining sensory inputs optimally in computer systems is a matter of research. In fact, it's probably the most central theme of research in multimedia computing. Especially, because in comparison to the effects observable in human beings (explained above) which must stem from rather elaborate computing methods in the brain, multimedia computing methods are very basic.

The subfield of multimedia computing that deals with sensor integration in computers is called synchronization. The remainder of this chapter therefore deals with synchronization. Another field that deals with sensor integration in machine learning is multimodal fusion, which is explained in Chapter XXX.

Introduction to Synchronization

Synchronization is not a new problem. Whenever there is coordination among two or more sources is required to complete a task, coordination among those is required. In many cases this coordination becomes establishing, specifying, and then performing this coordination in space and time precisely. Two common examples of these are orchestra and any team sports. In soccer or football two or more players must be in a particular spatial configuration and must perform their roles perfectly for desirable outcome. In music, space is not that important but multiple players must produce particular sounds at specific instants for effect. In fact the term orchestrate means "*Arrange or direct the elements of (a situation) to produce a desired effect*". A music conductor uses specifications in a music book to coordinate the timing by individual players. As we will see in this chapter, synchronizations is to specify timing and location of each multimedia stream to create a holistic impact.

A common first step in extracting information from multiple streams as well as presenting information for final consumption from multiple streams is to make sure that they represent the same event although from different perspective and maybe using different modality. Since in most cases, the data from different sensors is collected independently and is usually transmitted and stored using different channels, care needs to be taken that all the data is synchronized. The process of synchronization usually referes to processes that are required to make sure that events in a multimedia system operate in the same unison as they do in the real world. The relationships that exist in the real world among different media components should be captured and maintained for information extraction as well as proper rendering for human experience. As we will see, the most important relationship that needs to be maintained is time, but spatial relationships and content relationships also need to be maintained in many applications. In this chapter,

we will discuss how these synchronizations affect and different techniques that are used to ascertain that such relationships are maintained.

In the following we first discuss content synchronization, then spatial synchronization. After these two concepts, we discuss temporal synchronization in more depth. It must be mentioned that temporal synchronization is the main synchronization approach. Much of the efforts spent in multimedia synchronization are on temporal synchronization because time plays very important role in rendering time dependent media. As we will see, increasingly people present even time-independent media such as text or photos in a video environment, where time plays a key role.

Content Synchronization

In many applications different types of data sets are semantically or functionally related to each other and make sense only if they appear together. Appearing together may be in terms of space or time or both. The important relationship to be maintained is the content relationship and should be rendered such that it makes sense. A common example could be appearance of a figure, or in some cases a text box or even a slide, close to the concept that it is related to. Embedding of figures close to their citation is commonly used. Increasingly, people are developing approaches to embed slide presentations along with a scientific paper. In all these techniques the emphasis is that one must present contents that are related to each other somehow close to each other.

It is very common that researchers present their work in a research article as well as a slide presentation that they may have made at a professional meeting. Usually, these two constitute a dual view of the same work, often quite different from each other. Slides represent help grasp the work at a high level and research paper presents detailed arguments. Since these two modes of presentations constitute a dual view, further utility can be gained if the two media are synchronized. In such a synchronized fine-grained alignment between slides and document passages could be constructed and presented, allowing a user to view both the slides and the document simultaneously.

This joint presentation of slides and a document can be prepared by finding a suitable fine-grained synchronization between them. Such a synchronization may be formulated as a problem of aligning slides to a corresponding paragraph span in the document. This problem was formalized in the Slideseer system as document-to- presentation alignment:

“Given a slide presentation S consisting of slides s1 to sn and a document D consisting of text paragraphs d1 to dm, an alignment is a function f(s) = (x, y), mapping each slide to a contiguous

set of document paragraphs, starting at dx and ending at dy where x < = y, or to nil." The results of such an alignment are shown in Figure 1.

Of course, this is only an example of content synchronization. One could consider many situations, and the number of situations is increasing rapidly with increasingly availability and discovery methods on the Web. In most situations, the first step is to discover the content, then align it and finally present it.

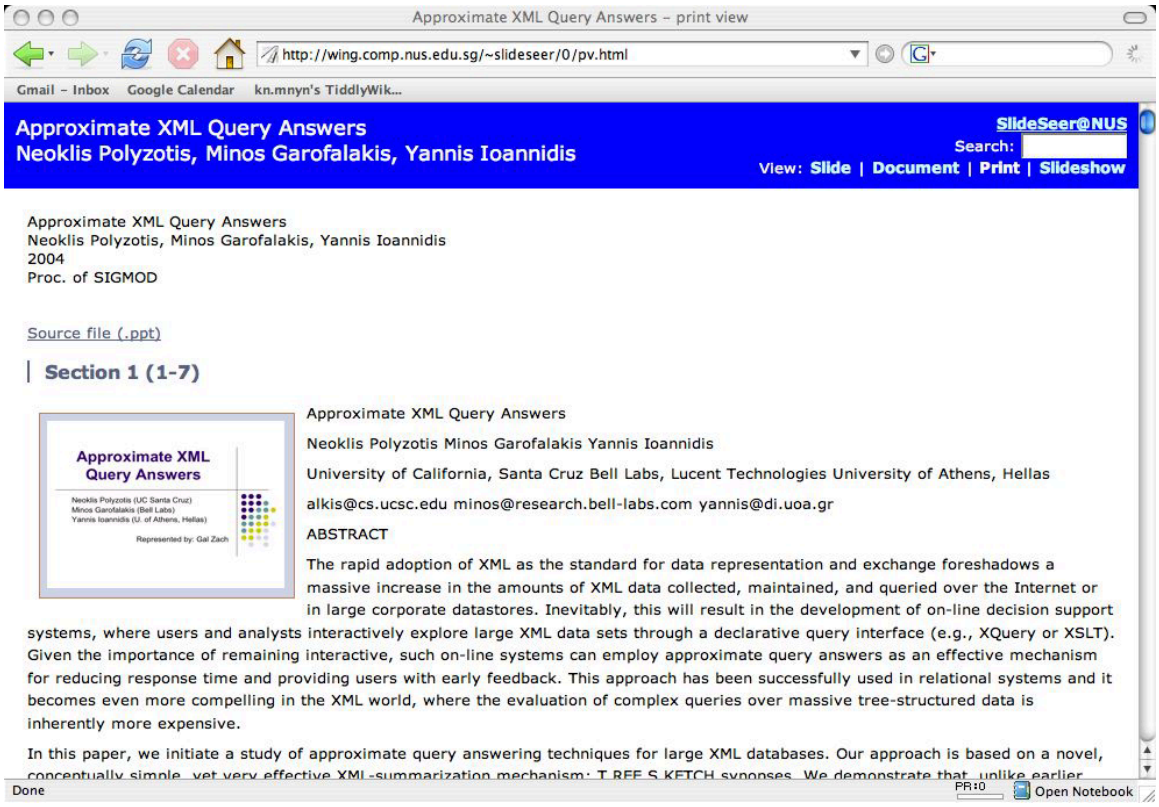


Figure 1: Alignment of slides and a research publication can be done analyzing contents of both and synchronizing them as reported in the Slideseer system [Reference].

Temporal Synchronization

Since many multimedia components are captured as time-varying signals and are also displayed in time, majority of the synchronization techniques have addressed issues in temporal synchronization. In this section, for brevity, we will drop 'temporal' unless needed in the context.

Synchronization techniques specify relationships that must be maintained among different media elements that must be interpreted or rendered together. In some cases all media elements are time-dependent, while in other cases such as making a video, some elements may not be time-

dependent, but must be rendered at a specific time.A time-dependent object is a stream in which each element has a specified time associated with it, while in a time-independent medium there may not be a stream (such as a photo or a text box) or the time relations may not be as critical, as in slide deck. Time-dependent object is presented or rendered as a media stream because there exists temporal relation between consecutive units of the stream. Time-independent object is the traditional medium such as text or images and could be rendered independent of time, except when they become part of a time-dependent media stream. Temporal dependencies among multiple media objects specify temporal relationships among objects that must be maintained for correct interpretation or understanding. A common example is the lip synchronization used in making movies. The image sequence stream showing a video must have a very precise temporal relationship with the audio steam representing speech for understanding what a person is saying. If the correct relationship is not maintained then a viewer may find the experience either poor or utterly confusing.

In many applications, temporal synchronization includes relations between time-dependent and time-independent objects as well. A slide show usually includes temporal synchronization of audio stream, either music or narration or both, which is time-dependent and individual slides which are time-independent media objects. In Figure 2, we show a time line and many different media objects that may be used to create a presentation or rendering of media as a function of time. For each media object its time duration is shown as a box on timeline. It is possible that at a given time multiple media objects may be rendered or at some times, no media object maybe rendered. The temporal relationships among time durations (commonly called time intervals) of different media objects may be specified using relationships first specified by Allen[XX]. Allen considered two processes and defined thirteen possible temporal relationships among them, see in Figure 3. These relationships have been used in specification of temporal relationship in many applications.

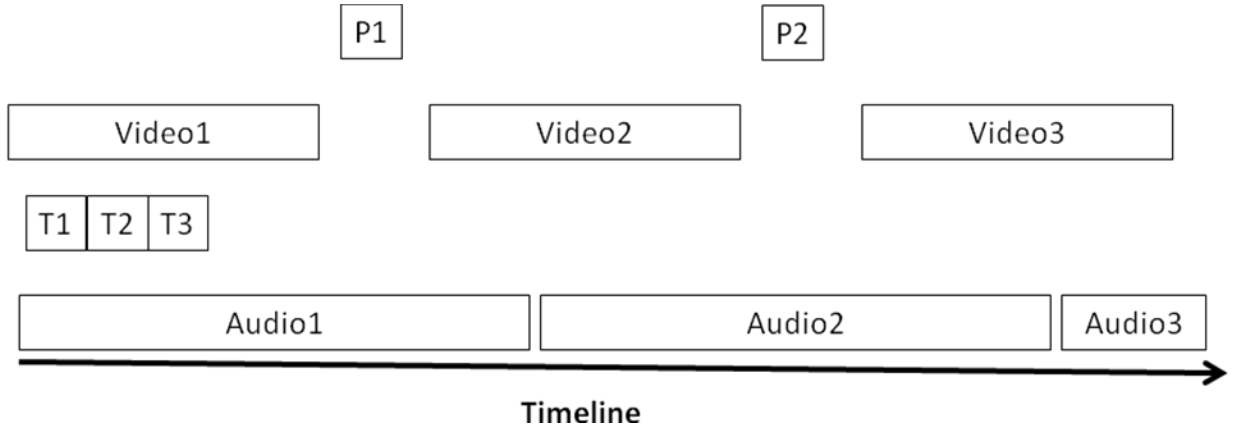


Figure 2: The duration of each media element, Audio, Video, text (T), and photo (P) are shown on the timeline.

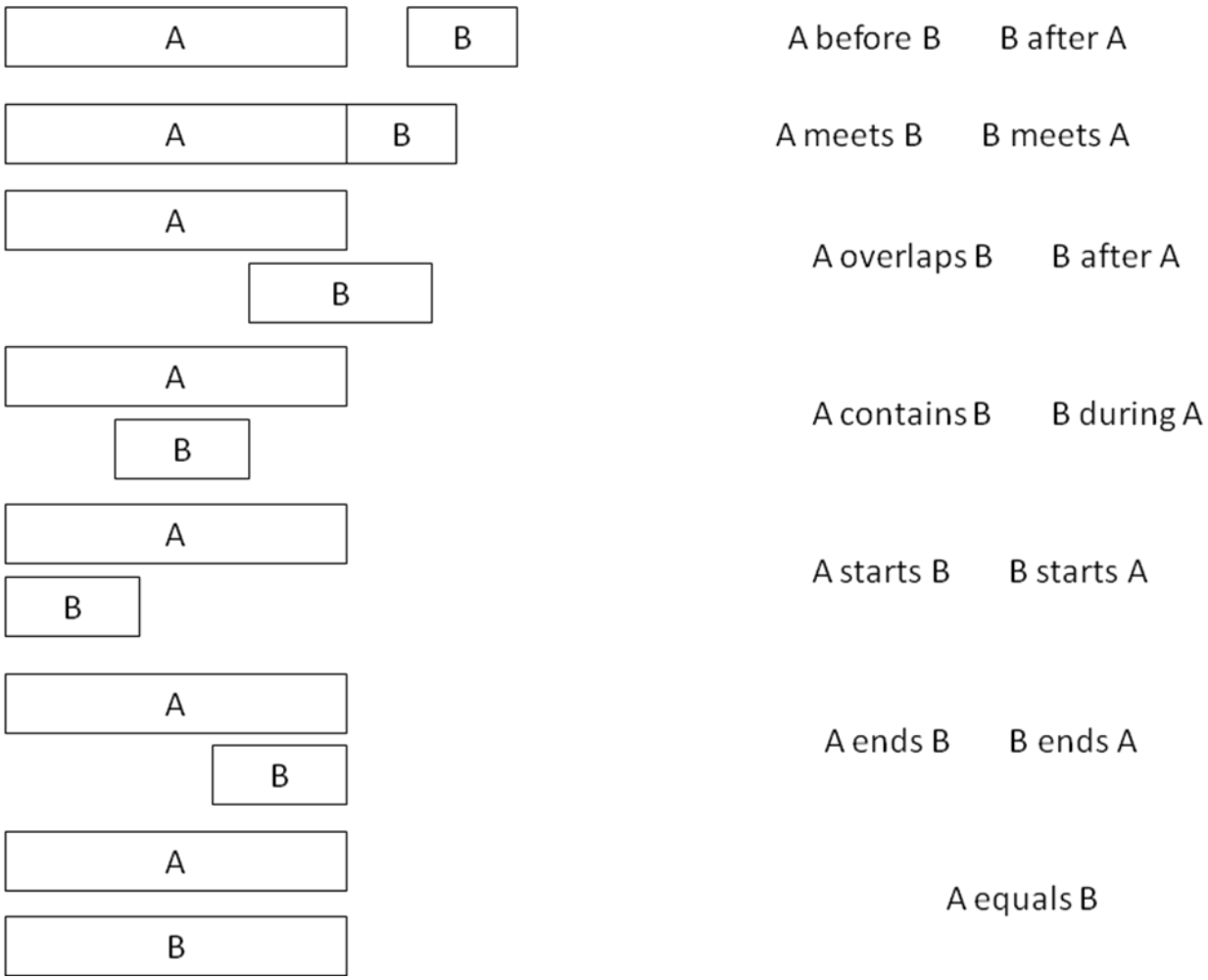


Figure 3: Thirteen temporal relations between two intervals A and B are shown here. Six of these are inverse of each other and hence there are only 7 relationships shown in this figure.

Synchronization Levels in a System

Synchronization has to be implemented at multiple levels in a system. We can consider three common levels of synchronization.

Level 1: At the lowest level, support must be provided to display each stream smoothly. Operating systems and lower communication layers are responsible for ascertaining smooth single stream by bundling and making sure about the display requirements. They make sure that there is minimal jitter at the presentation time of the steam.

Level 2: The next level is commonly called the *RUN-TIME* support for synchronization of multimedia streams (schedulers) and is implemented on top of level 1. This level makes sure that skews between various streams is bounded to acceptable limits dictated by applications.

Level 3: The top level deals with the run-time support for synchronization between time-dependent and time-independent media. This level is also responsible for handling of user interaction. The main responsibility of this level is to make sure that the skews between time-dependent and time-independent media are within acceptable limits.

Specification of Synchronization

There are many ways to specify synchronization requirements among different media objects. In this section we discuss different specification approaches that may be used in applications depending on the type of media used and application requirements.

Implicit specification: In many applications, temporal relations among different media components are determined implicitly during their capture. The goal of a presentation of these objects is to present media in the same way as they were originally captured. The relationships among media objects is considered specified implicitly at the capture time. A common example of this is the capture of a video that consists of two media components: audio stream and image frame stream. These two streams are captured by the system using the same time line. Thus we may consider that the synchronization requirements are specified implicitly.

Explicit specification: Temporal relation may be specified explicitly in the case of presentations that are composed of independently captured time-dependent objects such as audio or video, time-independent objects such as photo, and manually created objects such as text-boxes and slides. Similarly, in applications like a slide show a presentation designer selects appropriate slides, selects audio objects, and defines relationships between the audio presentation stream and slides for presentation. In such cases, the relationships among different media objects must be explicitly defined and specified by the designer.

Intra-object Specification: An animation video comprises of all manually created image frames that are presented at appropriate frame rate to create a video. In all such cases, thought there is only one media stream, the objects in each frame in the sequence must be drawn to create

specific visual effects. This requires that considering the motion characteristics to be conveyed, relationships and positions of different objects in each frame must be specified precisely. Thus, one needs to consider synchronization of the objects, their attributes, and their positions in each frame as a function of time in the presentation stream.

Deadlines

For synchronizing two streams, applications may specify whether their requirements for starting or ending media are rigid or flexible. In case of rigid requirements, the system must make sure that the deadlines are considered hard and must meet specification. Soft or flexible requirements mean that there is some tolerance specified within which the relationship must be maintained. For example, in case of speech related audio and corresponding video showing a person's face, it is important that audio and video are played within a specified interval.

Spatial Synchronization

Spatial placement of different components of a document have played important role. As we saw in multimedia authoring, relative placement of a photo and its caption is important and must be clearly specified. If a caption and its corresponding figure do not appear together then it may result in a significant confusion. In fact, in many cases the semantics of the document or a simple figure may completely depend on the relative placement of its component. In Figure 4a, we show many components in a random order. By placing them appropriately with respect to each other in a spatial configuration, we get Figure 4b, which represents a stick figure representing a person. Without spatial relationships among different components, we will not be able to render the message that we want to communicate. In fact, one can easily see that by a different spatial arrangement of components, one will get a different object.

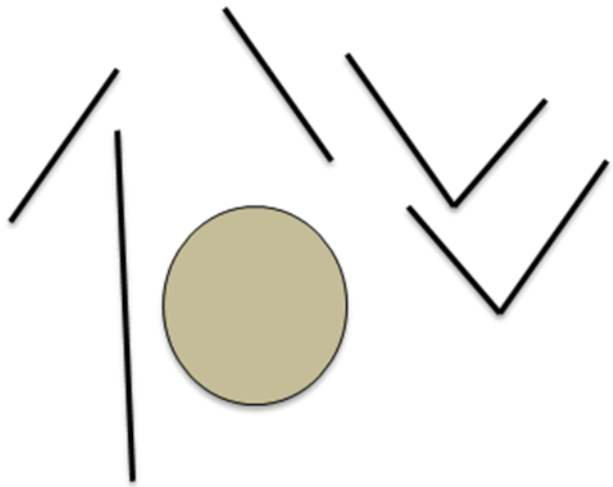


Figure 4a: Several components displayed in a random order.

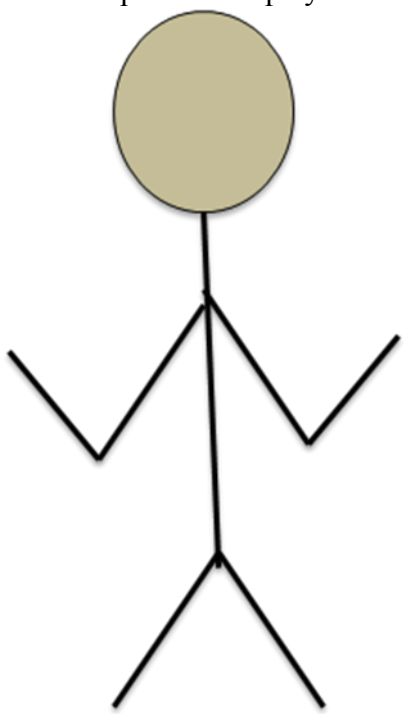


Figure 4b: The same components displayed in a specific order.

Rendering of different components of media is usually on a screen that is two dimensional. In some cases, one may want to consider rendering to take place in three dimensions, but in this chapter we will limit our discussion to two dimensional screens. On this screen, one must specify location of each media component as well as locations of sub components in each specific media.

A first step towards specifications is to define space used for presentation of a media object on an output device at a certain point of a time in a Multimedia presentation. For this, ***Layout frames*** are defined on an output device, usually a screen, and each content is assigned to be inside this frame. Positioning of the layout frame is usually the size of the screen used by the application. Within this frame, windows corresponding to different media (photos, video, or even audio symbols) to be displayed could be defined. In many systems, the layout frame usually corresponds to the complete screen and the positions of the other windows are then defined relative to the layout frame.

In defining relative layouts and even objects within each media windows, it is useful to define more than one coordinate system. For example in Figure 5, we have defined three coordinate systems. We have a coordinate system, F, defined for the layout frame. Each media window is defined inside this window so the objects within each window could be defined with respect to the coordinate system, W, defined for it. Finally, components of an object in a window could be defined in coordinate system, O, defined for a specific object in the window. By using multiple coordinate systems, and defining relationships among them, it is easier to define relationships that must be maintained with respect to different components.

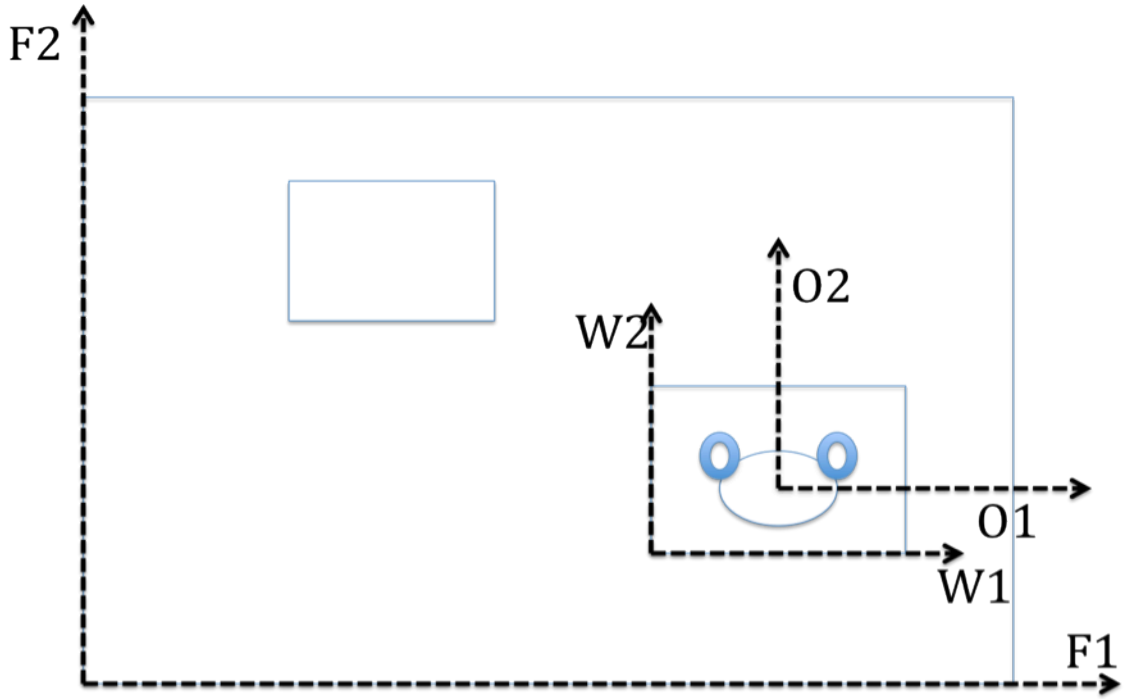


Figure 5: Use of multiple coordinate systems to specify positions among objects that must be maintained among them. The dashed lines represent three different coordinate systems F, W, and O, for Frame, Window, and Object, respectively.

Index Terms

Multimodal Interation, Split Attention, Cognitive overhead, McGurk Effect, Ventriloquism, Body Transfer Illusion, Synchronization, Synchronization - levels of, run time synchronization, layout frames, temporal synchronization, spatial synchronization

Literature

- C. A. Grimes, E. C. Dickey, and M. V. Pishko (2006), Encyclopedia of Sensors (10-Volume Set), American Scientific Publishers.
- Gemma Calvert, Charles Spence, Barry E. Stein: The handbook of multisensory processes, MIT Press, 2004.
- Baars, B. J. (1988): A Cognitive Theory of Consciousness. Cambridge University Press, Cambridge, UK, 1988.

Research Articles

- M. Hershenson: Reaction time as a measure of intersensory facilitation. J Exp Psychol, 63:289–93, 1962.

- H. McGurk and J. MacDonald: Hearing lips and seeing voices. Nature, 264(5588):746–48, 1976.

- Hahn, S. and Kramer, A. F. (1998). Further evidence for the division of attention among non-contiguous locations. Visual Cognition, 5(1-2):217–256.

- Narcisse P. Bichot and Kyle R. Cave and Harold Pashler (1999). Visual selection mediated by location: Feature-based selection of non-contiguous locations. Perception & Psychophysics, 61(3):403–423.

- Chandler, P. and Sweller., J. (1992). The Split Attention Effect as a Factor in the Design of Instruction. British Journal of Education Psychology, 62:233–246.

- Sweller, J., Chandler, P., Tierney, P., and Cooper, G. (1990). Cognitive Load as a Factor in the Structuring of Technical Material. Journal of Experimental Psychology: General, 119:176–192.

- Glowalla, U. (2004). Utility und Usability von E-Learning am Beispiel von Lecture-on-demand Anwendungen. Fortschritt-Berichte VDI, 22(16):603–621.

- Mertens, R., Friedland, G., and Krueger, M. (2006). To See or Not To See: Layout Constraints, the Split Attention Problem and their Implications for the Design of Web Lecture Interfaces. In Proceedings of the AACE E-Learn – World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, Honolulu, Hawai, USA.

- Botvinick, M. & Cohen, J. D. (1998). Rubber hand ‘feels’ what eyes see. Nature, 391, 756.

- Min-Yen Kan (2007) SlideSeer: A Digital Library of Aligned Document and Presentation Pairs, In Proceedings of the Joint Conference on Digital Libraries (JCDL '07). Vancouver, Canada,

June.

Exercises

5. Explain why the McGurk effect is generally not a problem for dubbing TV shows and movies into a different language.
6. Find at least one more example for the split attention issue in daily life. Propose a solution.

Chapter 9: Multimedia Systems

A multimedia system is a computing system designed to facilitate natural communication among people, i.e. communication on the basis of perceptually encoded data. Such a system may be used synchronously or asynchronously for remote communication, using remote presence, or for facilitating better communication in the same environment. These interactions could also be to communicate with people across different time periods or for communicating knowledge created over a long period. Video conferencing, video on demand, immersive video or immersive telepresence systems are all multimedia systems. Augmented reality or assisted communication systems, modern conferencing environments utilizing audio-visual mechanisms to facilitate communications are also multimedia systems. There are already systems that utilize touch or haptic communication. Efforts to communicate smell and taste are also making progress. The basic goal of a multimedia system is to communicate information and experiences by humans to other humans. Since humans sense the world using their five senses and communicate their experiences using these senses and their lingual abstractions of the world, a multimedia system should also use the same in communications.

Until late 20th century, communication and computing systems evolved separately. Communication systems usually dealt with the information or experience to be communicated in one sensory mode and converted it to some form of signal that was then transmitted to the recipient and converted back from the signal to a form that could be easily presented for consumption. As technology progressed, every sensory mode was converted to a common form – a digital sequence representing signal in sampled and quantized form that could be effectively and efficiently communicated over the same transmission medium independent of its original mode. As communication systems became digital, computing systems started playing an important role in different stages of communication systems.

Multimedia systems combine communication and computing systems. In multimedia systems, the notion of computing systems and communication systems basically becomes so intertwined that any efforts to distinguish them as computing and communications result in a difficult exercise. In this chapter, we discuss basic components of a multimedia system. Where appropriate, differences from a traditional computing system will be pointed out explicitly along with the associated challenges.

Components of a Multimedia System

Computing systems have evolved with advances in technology as well as requirements of users. At a very high level, it can still be considered the same and be used to understand major evolutionary changes in different aspects of the computing as multimedia started playing roles and started enforcing new requirements. The traditional computing architecture is shown in Figure CompArch. It consists of 4 basic components: Processing unit, storage, input, and output. In this section, we will consider the changes in this basic architecture as multimedia systems evolved along with other aspects of computing, and as we expect them to continue evolving in the near future. Multimedia systems have timing constraints. Different components, such as audio and video, must be rendered continuously maintaining specified time constraints as required by an application.

In a traditional computing system, most input represents an independent data stream that may have inherent temporal relationships within its data elements. Such data can be considered as one stream independent of any other data sources/streams. Thus, even in cases when data is coming from many sources, the system considers them independent sources and can process, send, and receive them using different networks. This is because the semantics of each stream is independent of other streams. Many networking approaches utilize this data characteristic as a fundamental assumption. As shown in Figure (Independent), these data streams are independent of each other and hence can be processed, transmitted in networks in packets, and assembled at the receiver independent of other streams. In Figure (dependent), we show data streams whose semantics is intertwined and efforts to consider them independently results in loss of semantics and hence information. This intertwining of data streams and joint semantics is what makes multimedia systems different and needs to be considered in most components of a multimedia systems. Some aspects and approaches for this intertwining and maintaining semantics of multimedia streams are discussed in the previous chapter on Synchronization.

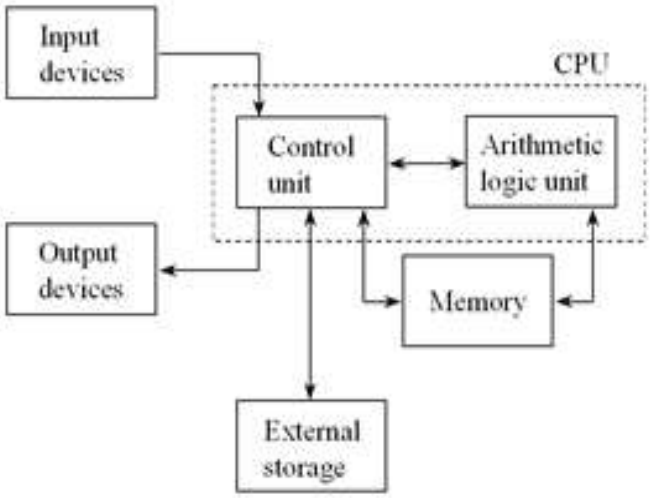


Figure CompArch (Redraw)

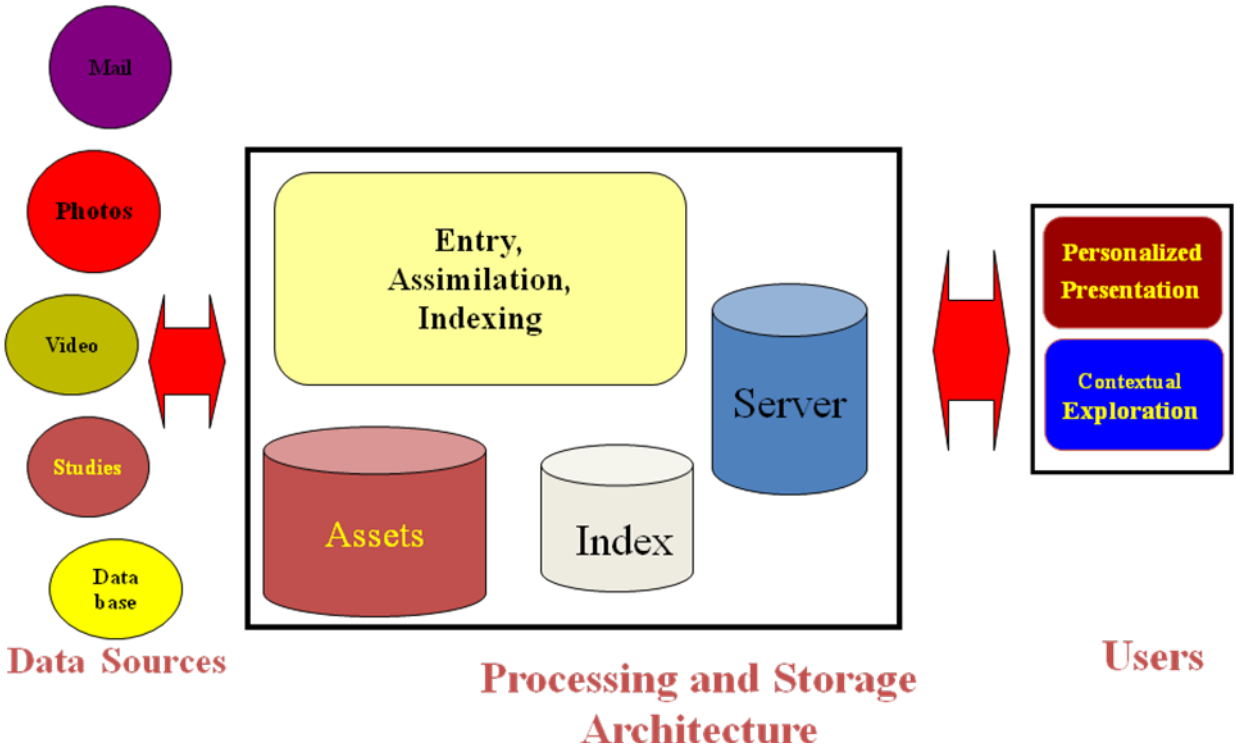


Figure Multimedia Architecture

Figure: Independent data streams.

Figure: Dependent data streams (relate this to figures in synchronization)

Processing Unit

The brain of a computer are its processing units. The fundamental processing units in a digital computer still remain the same, though they have become many orders of magnitude faster, smaller, and cheaper. A mobile phone today packs more processing power than the computers that occupied large air-conditioned rooms. Another major difference is that processing can be distributed using powerful networks so many, sometimes millions, processors located geographically at different places may function in concert as one processor. But underneath all these changes the constant is that the processing is still on binary numbers and builds complex operations using simple operations on binary numbers. All multimedia operations must be converted to these simple operations for execution.

Storage

Memory or storage is a fundamental component of computers that made it different from calculators and resulted in development of so many different applications. In the last six decades of digital computing many different types of storage mechanisms have evolved and continue to be evolving. Many levels of storage systems have been designed based on their relationships to the processing units. These are commonly called main (or primary) memory, secondary memory, and tertiary memory. Like processing units, the capacity of storage units at every level has seen enormous growth in its capacity and speed while equally dramatic reductions in its cost. Memories have now their own processing units to make them be more effective in communicating with the processing unit as well as input output devices. However, the fundamental nature and role of memory remains the same: store binary data. All multimedia data is ultimately converted to binary data along with its associated metadata that is also binary. The data management unit of the operating system is responsible for maintaining and using information related to the type of the data and how it should be transferred in and out of memory to other units.

Input/Output Devices

The most important difference between traditional computing systems and emerging multimedia systems is that the input data streams as coming from different devices in multimedia systems are strongly correlated and this correlation is strongly manifested by their spatio-temporal relationships. These relationships must be maintained within allowed variations otherwise the semantics or the quality of experience of these signals deteriorates rapidly leading to sometimes meaningless communication. In the chapter on synchronization (Chapter Synchronization) we

will discuss some of these aspects. In this chapter, we will discuss some computational aspects of these potential relationships and their effect in computing systems.

All multimedia input devices are sensors that capture physical measurements in an environment and convert it to a signal that is then converted to its digital form and handles in a system appropriately. This will be discussed in depth in the subsequent part of this book. In addition to the sensor inputs, metadata is often used to conserve the spatial and temporal relationships among various signals, as already discussed in Chapter XXX. Each signal type is then finally rendered in its type using a physical device. Thus, audio signals captured using a microphone, must be rendered using a speaker and visual signals captured using a camera must be rendered using a display device. The characteristics of the output device on which a signal is rendered should meet characteristics of the input device for reproducing the signal in the same way. We discussed some characteristics of these in other chapters. Details of these should be found from the manuals from the devices and are not discussed here.

Networking

Nowadays, multimedia systems are often based on intense utilization of networks. In traditional computing, computing used input from local devices and the output was also provided to local devices. With the popularity of PCs and the World Wide Web, communication started becoming a major component of computing and that synergistically expanded the role of networks in computing. Once communication started becoming popular, and multimedia devices started becoming affordable the nature of computing went through a major transformation. The first decade of this century saw major growth in audio-visual information, experiential knowledge, and audio-visual communication and entertainment. There has been a dramatic increase in the volume of multimedia information, as shown by the chart in Figure Growth of MM.

Multimedia information is naturally produced at one place and is edited, stored, and disseminated at other geographic locations. On the other hand, increasingly live communications rely on large volume of data transferred as High Fidelity and High Definition data across the world. To feed to this trend, most mobile phones are now becoming multimedia communication devices and are owned by more than 75% of the population of the world. All these suggest that in the near future, we are likely to see increasing use of multimedia communications in all aspects of human activities. This will replace the dominant medium – the text – until recently. This is a very important consideration in multimedia systems.

In current multimedia systems, input devices convert the analog signals commonly created in the physical world to digital signals and then these digital signals are appropriately processed and stored. Similarly, all digital signals must be converted to analog signals for rendering them in human consumable form. This is shown in Figure Comm. Since most data is processed and stored as digital signals, all multimedia communication is mediated by computing systems. In the figure, we show only two different nodes for input and output of the multimedia data streams. One must consider several such nodes and consider how these nodes are placed and how they are related in space and time to understand how multimedia system may be designed.

Different Configurations of Multimedia Nodes

Multimedia content is created and consumed in many different forms. The way content is created and then distributed, determines the architecture and technology that is

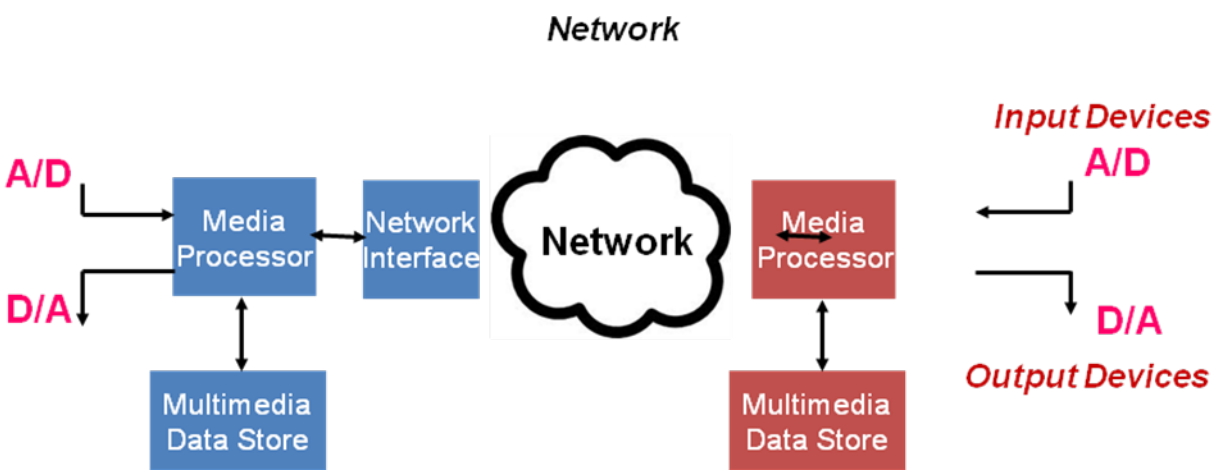


Figure Comm: Multimedia communication systems capture and render signals in analog form, but process and store it in digital forma. This makes computing systems an essential component of communication systems.

required. This is an area that is likely to be evolving for some time in the future. In this chapter we discuss some modes which have been commonly used for some time and some that are evolving rapidly and are likely to be common in near future. Before discussing different modes of multimedia consumption we briefly discuss the notion of *Quality of Service* (QoS) and *Quality of Experience* (QoE) that are used in quantitatively characterizing parameters to be used for measuring performance of the overall system as well as using these parameters for controlling resources in the system.

QoS and QoE

Quality of Service is commonly used to define and control performance of applications distributed over several computing related resources. For applications that consume many resources (including processing, storage, bandwidth, and I/O devices), techniques to ensure end-to-end service quality by managing resource utilization is essential. As we will see in the following, resource management and protocol implementation needs to consider the effect of decisions on the overall performance of the system as perceived by the user. QoS allows specification of the expected performance of the system by combining effects at different layers of the system. QoS specification must consider a multitude of properties including performance characteristics, availability, responsiveness, dependability, security, and Adaptivity of individual layers and components.

In developing approaches to specify QoS, one considers descriptions of quantitative *QoS parameters* (jitter, delay, and bandwidth) and qualitative QoS parameters, such as processor scheduling policy and error recovery mechanisms. *QoS specifications* must be declarative in nature. They should specify what is required without specifying how the requirement should be carried out. Finally, these specifications should be accompanied by a compilation process to map the specification to underlying system mechanisms and policies.

Quality of Experience (QoE) is related to but different from QoS. QoE is a subjective measure of a customer's experiences with a media as delivered by the system. Unlike QoS which focuses on the performance of the system and is measure in objective terms such as delays, use of bandwidth and jitter, Quality of Experience systems try to quantify what customer will directly perceive as a quality parameter. It usually asks questions about the performance in subjective terms.

Stored Video

In this mode a video stream has been acquired, edited and stored on a server. This video is now ready for distribution. The video data contains two major streams one containing an image sequence and the other containing audio stream. Both these may be already compressed and stored.with proper synchronization information. Depending on the location of the user, the copyright and other business issues, and the technical specifications of the server and client, the video may be made available in one of the possible two modes as discussed below.

Download and play

In this mode, the whole video is first downloaded on the client so it is stored there. And then it can be played. In this mode, the client requires enough storage, and the video is downloaded once and then can be played many times.

Streaming multimedia

Streaming multimedia usually means real time transmission of stored video such that it is rendered at the client maintaining synchronization among different components. In some cases, even live video from multiple locations could be combined into streaming multimedia components as we will see in a following section.

In streaming mode, the content to the client need not be fully downloaded before it starts rendering. While later parts of media components are being downloaded, the streaming player starts playing the media. In effect, the system maintains a buffer of the content by estimating network conditions so the application can display media to users while guaranteeing quality of experience.

In Figure **Streaming**, we show components of a streaming video system A streaming media system usually has the following components. We discuss these components considering the most common example of video streaming.

Compression: Most media is compressed, i.e. converted into in a more efficient but possible lossy representation before it is stored and transmitted. Compression techniques are discussed in Chapters XXX-XXX.

Application Layer QoS control: In networks, packet loss and congestion affect the timing and quality of streams received. Application layer QoS control is responsible for dealing with congestion on the network and maximize video quality even when packet loss takes place. Network congestion may result in bursty loss and excessive delays. Congestion control techniques attempt to minimize the effect of congestion by matching the transmission rate of the media stream to the projected bandwidth that may be available. These techniques usually try to control the rate based on source, receiver, or both.

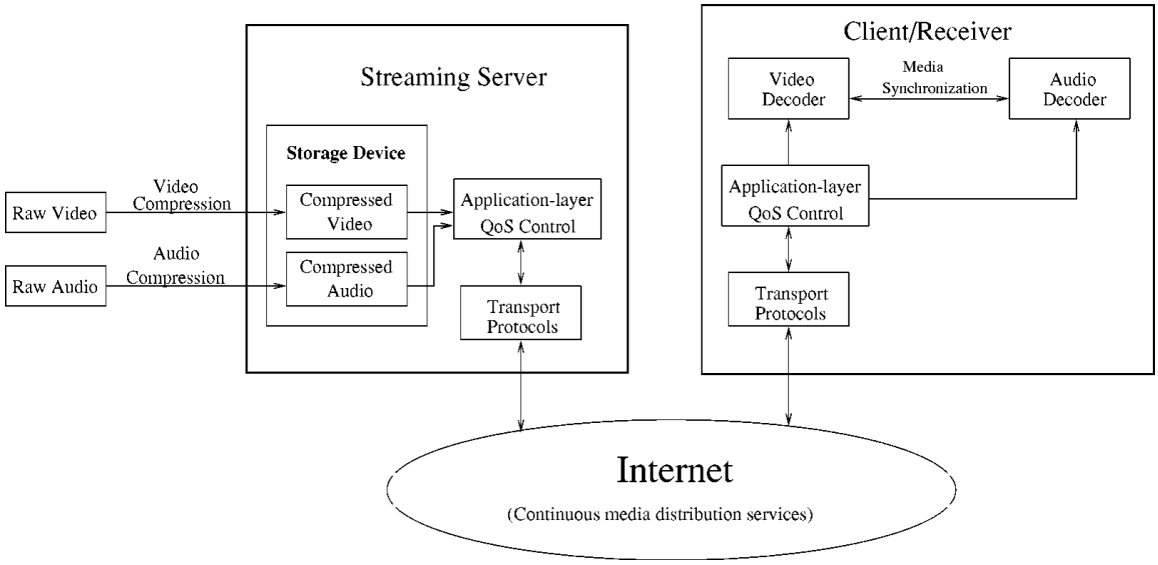


Figure Streaming (To be redrawn)

Continuous Media Distribution Services: Audio, video, and other media streams are considered continuous media because they represent a sequence of media values that are time ordered and make sense only in that order for the media as well as in relation to other media. Moreover, with respect to other media, synchronization is required to preserve the semantics as well as quality of experience. Since this media is transmitted over the best-effort Internet, using IP protocol, the sequence of received data may not be guaranteed to be the same as that transmitted. Many techniques have been developed to address some key problems in this area.

Network filtering techniques estimate network congestion and maximize video quality by adapting the rate of vide streams according to the network status. To reduce the load on the video server, such filters may be placed in the network. Server and clients usually have separate channels for control signals and for data to the filter.

Another technique used for media distribution is *IP Multicast*. Normal networks deliver content or data to only one client. IP Multicast is designed to deliver the content to multiple clients as dictated by the application. Such a service is built on top of the internet protocol. This allows application providers to build their multicast services on top of the regular Internet.

Content replication is used for improving scalability of media delivery systems. Two common approaches for this are caching and mirroring. These are used for reducing the load on the streaming servers, reducing bandwidth requirements, improving latency for clients, and for increasing availability to more clients. In mirroring, a copy of the original content is placed at multiple locations in the network, based on estimated requests for it. This means that at a par-

ticular location, multiple users will be directed to the copy that is locally stored. This reduces bandwidth requirement as well as latency. Caching of the content for a client is done based on estimation of the use of specific content by them.

Streaming Servers: Streaming servers process multimedia data under timing constraints to prevent undesirable artifacts during presentation at the clients. They are also responsible for providing stop, pause/resume, fast forward, and fast backward operations when requested by a client. Finally, retrieving media components and presenting them while guaranteeing synchronization requirements is also their responsibility. Streaming servers accomplish these by using three main components: Communicator, Operating System, and Storage System. As shown in Figure <Streaming>, the communicator involves the application layer and transport protocols. A client communicates with a server and receives multimedia contents in continuously, while guranteeing synchronization.

The operating system requirements in multimedia systems are significantly different. An operating system for streaming services must satisfy real-time requirements for streaming applications. A storage system for streaming services has to consider the continuous nature of media components and support storage and retrieval to meet those requirements.

Like an operating system in a general computing system, a real time operating system is responsible for controlling and coordinating all resources in computing environment by considering all requests and managing and scheduling all resources including processors, main memory, storage, and input/output devices. The major new requirement is to mange additional requirements resulting from the large volumes of continuous data that also has synchronization and timing requirements. The added requirements in real time operating systems in multimedia computing are addressed by considering process management for addressing the timing requirements of streaming media, resource management for meeting timing requirements, and file management to address storage management.

Storage management for multimedia data becomes challenging due to the large volume of data, high throughput, synchronization requirements, and fault-tolerance. One of the most challenging issues has been related to high throughput requirements for the data while maintaining synchronization among different streams. Three commonly explored approaches have been: data stripping, disk-based storage, and hierarchical storage. These three approaches are shown in Figure<storage>. Data stripping techniques are used to allow multiple users to access the same video content at high throughput. A video is split into many segments and these segments are scattered on multiple disks, which can be accessed in parallel. This scheme is shown in Figure

<Storage> a. This scheme allows multiple users to access different segments in parallel enabling high throughput. The most obvious scheme is shown in Figure b. In this scheme, multiple disk arrays are used to manage large volume of video data. These disk arrays can become really very large and expensive. Cost considerations result in the hierarchical storage architecture, shown in Figure c. As seen here, now one uses three levels of storage: memory, primary storage commonly in the form of disks and disk arrays and retiary memory in the form of tape library. For even larger scale deployment of content services, storage area networks comprising of high speed data pipes and multiple sorage centers are implemented.

Media Synchronization: The semantics as well as the quality of experience of multimedia depends on ascertaining proper synchronization among different streams. Synchronization becomes challenging due to the nature of current networks and protocols designed for traditional data streams. Considering importance of synchronization, this issue is discussed in details in the chapter <synchronization>.

Protocols for Streaming Media: Communication between clients and streaming servers is specified at three different levels: Network-layer, transport, and session control. In Figure <protocol> we show relationships among different components and the protocols. For addressing and basic services in the network, IP used commonly in Internet services is also used by multimedia services.

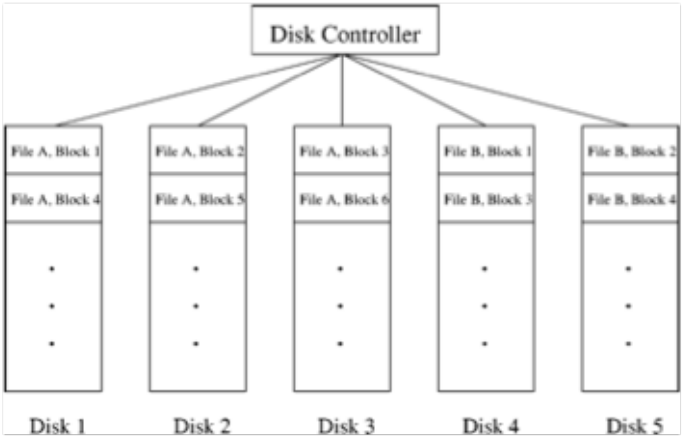
Transport protocols, such as UDP (user datagram protocol) and TCP (transmission control protocol) are used at the lower layer for basic transport of data. RTP (real time transport protocol) and RTCP (real time control protocol) are implemented on top of those basic data protocols and are used to manage multimedia communication.

During a session between a client and a stream server, all messages and control communications use session control protocols. RTSP (real time streaming protocol) is used for establishing and controlling media streaming sessions. This protocol is responsible for specifying and performing VCR like operations. SIP (session initiation protocol) is used for creating modifying, or terminating two party or multiparty communications using multiple media streams.

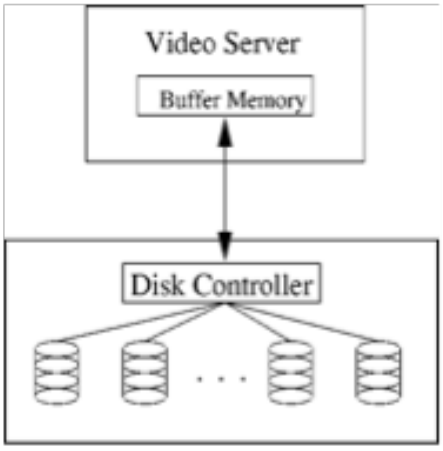
As shown in Figure <protocol>, the compressed media data streams at RTPlayer, where RTP – packetized streams are assigned sequence numbers as well associate synchronization information. These packetized streams are then passed to UDP/TCP layers, which in turn pass this to IP layer for transporting to Internet. The control signals using RTCP and RTSP pacts are multi-

plexed and move to the IP layer for transport over internet. At the receiver the signals move in the opposite direction, from IP to data plane for rendering on appropriate devices.

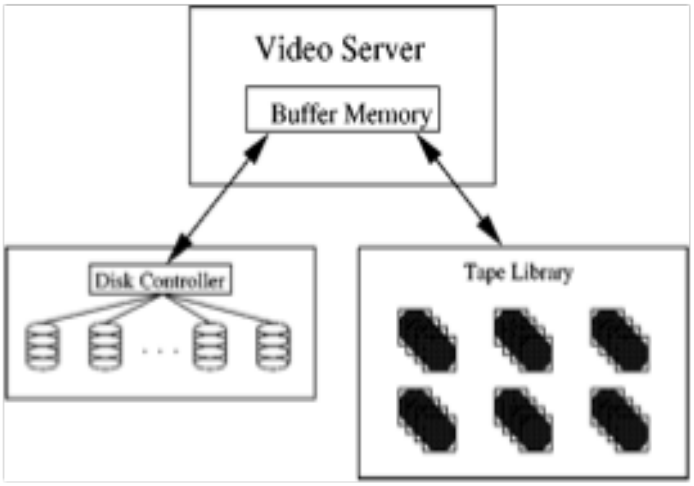
Figure Growth of MM.



a. Data stripping



b. Disk-based storage architecture



c. Hierarchical storage architecture

Figure <Storage – to be redrawn>: Three different architectures are used in video servers for storage management.

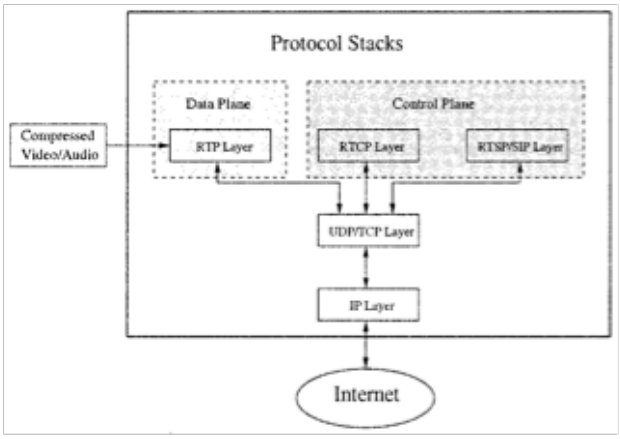


Figure <Protocols> <Redraw>

Live Video

Video is the most common form of streaming media. A video usually combines at least two continuous media: audio and image sequences. In many applications, text streams may also be associated. Recently in many applications, tactile and other media are also being combined in special

applications of video. In this section some established modes of video are discussed along with some emerging forms of video. It is important to see how different types of streams are prepared and consumed to design architectures for these systems. It is expected that this area will remain very active and with advances in technology, many new modes and applications will emerge. However, in this section we will discuss only those modes that have been active already for some time.

One camera video

A significant fraction of video applications used video that is captured using only one video camera. Such video usually has single image and audio streams and both those streams are captured using closely located camera and microphone. During the editing process some other streams may be added, but in this case synchronization issues are the easiest and video can be easily transmitted using common video transfer or streaming techniques.

Multiple camera videos/produced as single

Most sports and entertainment videos are captured using multiple cameras and then edited and produced into a single video stream. If the video streams are edited just select cameras in a particular segment then synchronization issues remain easy. If one selects image stream from one camera and audio stream from another camera or an independent microphone, then in the editing phase one must consider relative location of image and audio source to synchronize them to make them natural. In this case, the editor has to take care of synchronization issues during the editing phase. Once the video is produced, it becomes a single video stream that is

Multiple Perspectives Interactive Video

In the traditional model of television and video is based on a single video stream transmitted to a viewer. A viewer has option to watch, and use standard VCR controls in recorded video, but little else. In many applications, such as in sports, several cameras are used to capture the event. A human producer decides which video stream should be transmitted to users at any given time in the event. Thus, the producer acts like a multiplexer deciding the stream that is transmitted while blocking all other streams. It is well known that different people may be interested in different perspectives of the event and hence may consider different sequencing of events as more desirable. Multiple Perspective Interactive Video, or MPI Video, can provide true interactivity to viewers. A viewer could view an event from multiple perspectives, even based on the content of the events. MPI video will overcome several limitations of the conventional video and provide

interactivity essential in applications ranging from scientific inventions to entertainment. In the conventional video, viewers are passive; all they can do is to control the flow of video by pressing buttons such as play, pause, fast forward or fast reverse. These controls essentially provide you only one choice for a particular segment of video: you can see it or skip it. In the case of TV broadcast, viewers have essentially no control. If a viewer must see what the broadcasters decide to show them or change away from that station. Even in those sports and other events where multiple cameras are used, a viewer has no choice except the obvious one of keeping the channel or using the remote control and go channel surfing.

A MPI Video system integrates a variety of visual computing operations along with three-dimensional modeling and visualization techniques to provide automatic analysis and interactive access to data coming from multiple cameras concurrently monitoring an environment. The system creates a three-dimensional model of dynamic objects, e.g. people and vehicles, within this environment, and provides tracking and data management of these objects. Using this information, the system supports user queries regarding the content of the three-dimensional scene. Such a system is shown in Figure MPI.

In MPI systems, the system knows the location of various cameras and using processing finds locations of objects of interest at every time instant. Based on user queries, such a system should then create a synamic sequence of vide based on a users request. Since there could be many users, each with a different request, the system must prepare as many different video sequences as different queries.

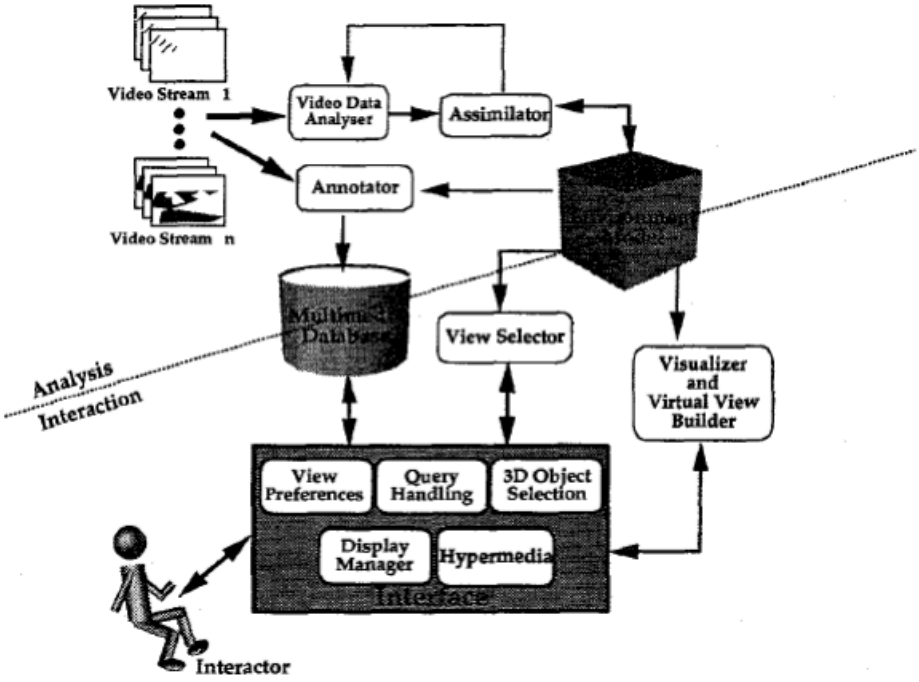


Figure MPI: (Redraw)

Immersive Video

Immersive video is a further enhanced version of MPI video. In immersive video, one uses visual analysis and reconstruction techniques to create a complete three dimensional model of the environment being observed by the cameras. This model contains all dynamic objects and has their models also in the system. Since the system has complete three dimensional model of the environment, it can recreate the scene from any perspective that user wants. This may allow a user to imagine that he is standing on a football field where players are in action and is observing all actions as in a virtual reality system. In fact, these systems could be called real reality system because they create the model of a real world to allow a user to experience this world in its complete visual reality. Such a system is shown in Figure reality and an example of what a user will experience is shown in figure real reality experience.

The high-level software architecture of an immersive video system contains the following

components:

1. A video data analyzer detects objects from each individual input video.

2. An environment model builder combines a priori knowledge of the scene with information obtained by the video data analyzers to form a comprehensive model of the dynamic environment.

3. A visualizer generates a realistic, virtual video sequence as interactively requested by the user.

Figure Immersive Video.

Emerging Systems

Multimedia systems are going through rapid evolution. Just two decades ago, most of the computing used only alpha-numeric data and hence all systems were designed to deal with very limited data types and did not consider continuous media in their design considerations. Today, in the early part of the second decade in the twenty first century the main computing and communication client is emerging to be a 'smart phone' which has more media power, in terms of number of sensors, than human beings and most communications are becoming multimedia contextual experience sharing rather than abstract textual sharing. It is difficult to imagine how we will share our experiences with people and how will we create knowledge in future. What is clear is that the evolution of this trend is going to continue for some time. In this section, we presented fundamental nature of multimedia systems. To understand details, one must approach vast literature in this area.

The paper by Wu et al, for example is a very rich resource for concepts as well related sources related to streaming media. We strongly recommend this paper for anybody interested in details in this area.

Index Terms

Literature

Research Articles

Allen, J.F., “Maintaining Knowledge about Temporal Intervals,” Comm. of the ACM, November 1983, Vol. 26, No. 11, pp. 832-843.

Min-Yen Kan (2007) SlideSeer: A Digital Library of Aligned Document and Presentation Pairs, In Proceedings of the Joint Conference on Digital Libraries (JCDL '07). Vancouver, Canada, June.

Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang, and Jon M. Peha, “Streaming Video over the Internet: Approaches and Directions”, IEEE Transactions on Circuits and Systems for Video Technology, VOL. 11, NO. 3, MARCH 2001

Jingwen Jin and Klara Nahrstedt, “QoS Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy”, IEEE Multimedia, July 2004.

Greg Thompson, and Yih-Farn Robin Chen, “IPTV *Reinventing Television in the Internet Age”, IEEE Internet Computing, May 2009*

A. Katkere, S. Moezzi, D. Kuramura, P. Kelly, and R. Jain, “Towards Video-Based Immersive Environments,” *ACM-Springer Multimedia Systems Journal, Special Issue on Multimedia and Multisensory Virtual Worlds* Spring 1996.

S. Moezzi, A. Katkere, D. Kuramura, and R. Jain, “An Emerging Medium, Interactive Three-Dimensional Digital Video,” *IEEE Multimedia*, June 1996

Kanade, T.; Rander, P.; Narayanan, P.J.; “Virtualized reality: constructing virtual worlds from real scenes “, in IEEE Multimedia, Jan 1997.

Zhenyu Yang, Wanmin Wu, K. Nahrstedt, G. Kurillo and R. Bajcsy: “Enabling Multi-party 3D Tele-immersive Environments with ViewCast”, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), vol. 6, 2010.

Web Links

Exercises

Chapter 10: Brief Introduction to User Interface Design

So far, we have mostly described ideal and typical environments. In this chapter, however, we present concrete advice and rules that will help build multimedia applications that provide a good experience to the user. The examples are mostly based on the GUI standards at the time of writing this book. Ideally, many issues will not arise anymore when the typical point and click GUI has been replaced by applications with immersive sensor experience. However, in the mean time: How can we make our multimedia application friendly to its human user?

There is no easy answer to this question. In this chapter, we will focus on a few important aspects of human-computer interaction (HCI), which we think provide enough introduction for the reader to be able to start thinking about the major issues and to continue studying the field further. We start with some general rules of user interface development and then continue with an introduction to human-based evaluation.

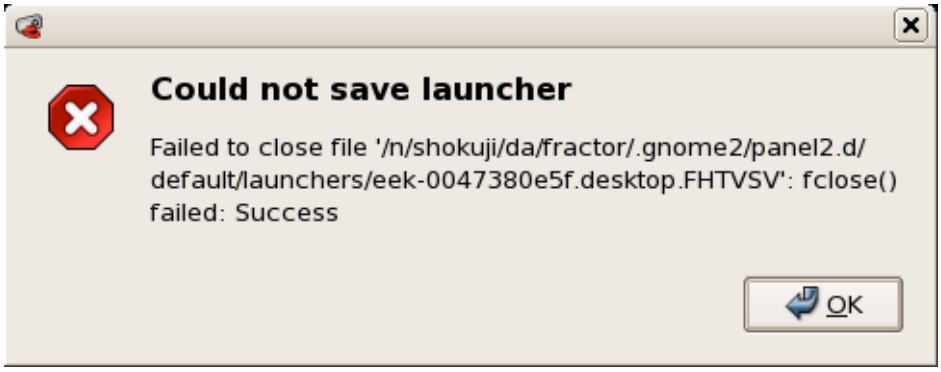


Figure 1. The developers of an application thinking inside-out is one of the most frequent causes of bad user interface design. This real-world example of a confusing dialog box presented itself to the authors during the creation of this chapter.

User Interface Design

Most of today's applications, especially ones that support multimedia in any way, use a graphical user interface (GUI), i.e. an interface that is controlled through clicks, touch and/or gestures and allows for the display of arbitrary image and video data. Therefore knowing how to design GUI-based applications in a user friendly manner is an important skill for everybody working in multimedia computing. Unfortunately, with many factors that go into the behavior of a program and the perceptual requirements of the user, there is no unique path or definite set of guidelines to follow. So is it better to have the menu bar inside the window of an application or is it better to

have one menu bar that is always at the same place and changes with the application?[6] As we assume the reader knows, this is one fundamental difference between Apple's and Microsofts operating systems -- and it is hard to say one or the other is right or wrong. However, some standards have evolved ver many years and using research results and feedback from many users. These standards can be seen in many places today in desktop environments, smartphones, DVD players, and other devices.

Don't think inside out: There is one important rule that it is universally true for any machine or application interface: Never assume that the user knows what is happening inside the program. This is called "thinking inside out". The interface should assume no knowledge of any kind of he insides of the program. While this seems obvious, it is more than easy to accidentally require much more knowledge about a program than a user has because, after all, the creator of the program has intimate expert knowledge. Most user interface glitches are the result of a violation of this rule. When we introduce ourselves to people, we start with stating our name and then go on to a small selection of facts that are relevant in the context of the conversation. A program should follow the same guideline: It should present itself inside the user interface conventions of the particular device and then only expose those details that are required and expected by the user. If the program "introduces" itself with difficult technical details that are hard to understand, the user, especially a first-time user will probably reduce his or her interaction to finding a way to end the program and think about a different solution to his or her problem. The dialog box in Figure 1, from a modern, heavily-used application, shows a bad real-world example of a developer thinking inside-out.

Not all issues are as obvious as the one shown in Figure 1. In fact, most of them are not, however, Figure 1 shows you how severe the issue can become.

[6] This particular example has been studied quite thoroughly, see the references for details.

Figure 2. An "OK" button in a normal dialog box usually suggest everything went fine. In this case a process failed, however, so the user should be presented a warning message.

Develop task-oriented rather than technology-oriented interfaces: This is one rule that especially true for multimedia applications. In multimedia applications designers tend to present users with as many outputs (that is, different media) as possible. This is often confusing to the user, who would find the system easier to use if there were fewer choices. Choosing wisely is key here. Most importantly, one has to focus on what the task is and what the user is capable of once the technology fulfills all requirements. Larry Wall, the creator of the PERL programming language is often quoted saying: "Common things should be easy, advanced things should be at least possible". Especially for multimedia applications this means having the function in mind first and then the presentation.

A common example that demonstrates this is: Just because you have several hundred fonts to choose from it doesn't mean a document should use all of them. In fact, typical publications only use 1 to 3 fonts because most readers would not find it appealing to look at a mess of fonts. Figure 2 shows a more subtle example of this problem: A user is presented with a dialog box when a problem occurred. Technology-wise it is easiest to think of any warning message as a dialog box. Task-wise this could result in disappointment and frustration for the user as an "OK" button in a normal dialog box usually suggest everything went fine. In this case a process failed, however, so the user should be presented a warning message.

In general, when designing the user interface to your application you should ask yourself the following questions:

- What are the (mental and physical) capabilities of the user, e.g. what are the technical terms understood by the user?

- What are the tasks that the program is helping the user to achieve and what information has to be presented and asked from the user for doing so?

-What are the upper and lower limits for resource requirements needed to accomplish the tasks, e.g. how much disk space is required and how much memory has to be allocated? A typical eror made here is to restrict array lengths arbitrarily and exposing the limitation to the user. Many of

the Y2K bugs fall into this category, as developers underestimated how long their application would be used. In the multimedia community, screen resolutions and sampling rates have often been part of this problem.

- How can the application make the most important tasks easy and quick and the less frequent tasks at least possible? In other words: Find the right compromise between the ease-of-use and the complexity of the program.

The most common issue for multimedia computing in this regard is probably a device that everybody has at home in abundance: The remote control. Remote controls are inherently technology oriented, especially multi-device remote controls which can be used to command TVs, DVRs, and home theater systems alike. While some of the buttons, such as volume control, program up and down, and play and pause show standardized symbols, most other buttons look differently on every type of remote control, and most users will agree that the function of some buttons remains a mystery forever. A task-oriented remote control, would only show buttons that are interesting to the user at a given point (e.g. when the DVR is used to playback a recording, the program buttons could be grayed-out). Also buttons that are rarely used should be hidden in an advanced setting. Of course, this is harder to realize with a physical remote control but it is possible in a display-based remote, e.g. on a cell-phone (see exercise #?).

Don't be creative when you shouldn't: Suppose you meet with your friends to play soccer. You have been doing this together for a long time, but this time you decide to change the rules without telling them. Small children know that such a behavior will most likely lead to confusion, frustration, and conflict. The same applies to user interface design. When the device or operating system manufacturer sets a standard for certain GUI elements to behave a certain way, your program should not change the behavior. You must understand the rules for a certain GUI element (such as a modal dialog box) and not use the element in a different way. Like the soccer example, doing so will lead to confusion and frustration of the user and also potentially to conflict with other applications that share GUI elements. Some device manufacturers also enforce GUI standards and will likely cause you to redo your interface during quality assurance. While this rule is true for any application, multimedia applications are the most prone to tempt you to abuse predefined UI components. The reason for this is that conventional operating system interfaces are originally created for text and mouse input and output while multimedia applications might deal with various inputs and require content-based selection and editing operations. Figure 3 shows an example of a system that uses a chalkboard metaphor instead of a desktop metaphor. As of the writing of this book, most of these are not yet standardized. As a result many requirements of the

Desktop metaphor had to be ignored when designing the system and replaced by rules that make more sense. The recommendation here is to stay as close as possible to the interface standard and only takes as much freedom as is required. Finding the right and intuitive solution might require user interface studies, as explained later in this chapter. Another firm rule here is that the user should never have to perform unnatural or unintuitive actions.

Figure 3. The electronic chalkboard system E-Chalk is an example of a multimedia system that would not work with the desktop metaphor.

Foster the learning process of the user: One of your goals as an application developer should be to bind the user to your program for a long time. This is best achieved when the user incorporates your application more and more into daily live and thinks of it increasingly often when a new problem comes up. A user's first use of any program is likely to be something simple but each task should increase the user's understanding of the your program so that the user gradually learns to use the application to solve more sophisticated problems. In this way it is very likely that the application will become an important tool in a user's life.

To foster the learning process, your application interface should follow one concrete philosophy (see example later in the chapter). For example, many operating systems follow the Desktop metaphor, therefore using terms like "folder", "file", and/or "trash bin". Sticking to the metaphor or philosophy is very important, inconsistencies or contradictions should be avoided. Most importantly, the environment should be made riskless along the philosphy. For example: Like in real life, things put in the trash bin, can also taken out of it when they haven't been in the bin for

too long. Undo and redo functionality make using computers for typewriting even less risky than using a physical typewriter. Riskless environments promote learning because making a mistake is not penalized heavily encouraging exploration. Another examples is intelligent default values: They help exploration because the user can see what happens before understanding what the values mean. Figure 4 shows another "bad" real world-example: even an experienced user would have to think twice about whether to say "Yes" or "Cancel".

Another property of a program that promotes learning is transparency. Of course, the user does not always have to know what is going on -- this would contradict the first paradigm of not thinking inside out. However, the user should always know what is expected of him and what he can expect from the computer. So if a certain action triggers an LED to light up, it should always do that. Programs should also not start tasks on their own without educating the user about it. Numerous system tools, such as virus scanners, have violated that rule in the past, by starting background or update processes with the result that the user becomes frustrated because the computer is slower or unresponsive without the user causing any action. A typical multimedia example here is fast-forwarding or rewinding of a video using a typical Internet player interface: Often it is not clear how long the process will take because it depends on the encoding of the video and the bandwidth of the connection between video server and player client. So many users don't even want to touch the controls anymore once a video has started playing. This issue is called *responsiveness*.

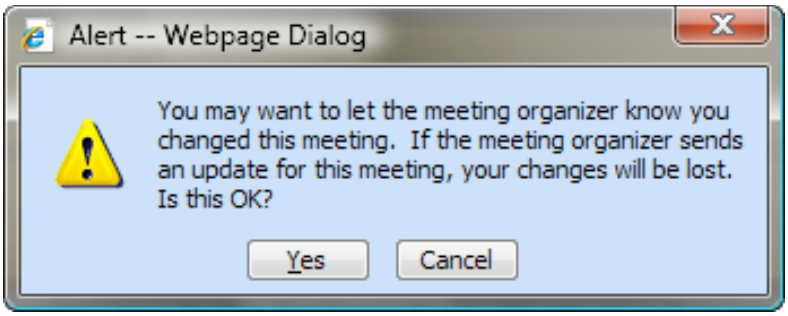


Figure 4. Fostering the learning process the wrong way: Even an experienced user will have to think twice to know what button to press.

Responsiveness refers to the ability of a program to complete a task within a given time. The paradigm of transparency requires to educate a user about tasks that will take longer than an expected time span. For example, when the computer is not responsive due to a compute or I/O-intensive operation, a busy cursor should be displayed, otherwise the user will think something is

wrong. If the operation takes longer than a while the operation should be cancelable and a progress indicator should indicate the estimated time until completion of the operation.

Responsiveness has been studied, especially in connection with multimedia computing (see references XXX). The general rule is that when a program is not able to respond to user interaction for more than about one to three seconds (depending on the general responsiveness of the device) the user should see an indicator for this being normal, e.g. an hourglass cursor. Other rules of thumb include the following: If an operation occupies the machine for more than about ten seconds, the application should show a progress bar. Operations of 20 seconds or longer or when the duration is hard to determine should show a progress indicator and should also be easily cancelable. Operations that take several minutes, e.g. the conversion of videos from one format to another, should inform the user of this and prompt confirmation.

Sometimes it can be tricky to assess the time duration of an operation because the computation time to perform a task of course varies from computer to computer, depends on the data being processed and may change with available bandwidth. One solution to the problem is sometimes to perform a small subset of the task (e.g. convert only a couple seconds of a video) and then estimate how long the completion of the entire task will take. It is usually acceptable to slightly overestimate. Another solution might be to perform the task in the background (e.g. on a secondary CPU core) and have the computer not be unresponsive to user interaction. Rather than disabling all the interaction, only interaction that depends on the task being completed is disabled. For example, further operation on a video being converted might be prevented but loading a second video might still be enabled. So the user may choose to perform a different task instead of waiting. Of course, it must be transparent to the user what the computer is occupied with and what interaction is still possible. A third solution, that works particularly well with many multimedia applications is to visualize the progress in addition to showing a progress bar. For example, when transforming a video, the program could show the frame currently being processed. While this does not cut down on processing time, in fact it will most likely increase processing time, it might cur down on perceived processing time as the user is engaged in the visualization of the content. As explained above, user interface design is not about objective numbers, it is about subjective satisfaction.

Test your program with real users: The most important rule is to always test your application with real users. The above stated principles and rules are neither comprehensive nor do they guarantee success of an application. Furthermore, let's assume you think your program presents

itself in a learning-promoting way and hides exactly the right amount of details. How can you be sure? After all, you might be thinking inside-out.

The rest of this chapter will introduce the basics of human-based evaluation.

Evaluation of Software through Human Subjects

Dix et al. (see references) formulated a simple equation for the measurement of the usability of a program:

$$Usability = Effectiveness + Efficiency + Satisfaction$$

Of course, quantifying the parameters of the equation is not only complicated but also subjective. Different parameters might have different importance in different applications. In a computer game, for example, satisfaction probably equals effectiveness. Depending on the game, efficiency might or might not matter. So how does one measure the usability of an application?

Again, there is no silver bullet method to measure the usability of an application. An often used method in industry is to equate user satisfaction with profit. For example, one of two webpage designs is shown to new visitors at random and then the company appearance is further optimized in the direction of the page that results in most sales. The method is called "split test". The main drawback of the method is that sales numbers might correlate with many other factors and decreasing sales numbers might not be caused by bad webpage design. Two more verifiable methods are the video surveillance tests and questionnaires. Of the two, video surveillance tests are probably the most effective.

Video Surveillance Test

For a video surveillance test, candidates are chosen that represent the typical users of a particular application. It is best to select persons to represent a variety in gender, age, ethical, social, and educational background. The candidates are then given a set of tasks to solve with the application. Sometimes it can be effective to also set a time limit. With the consent of the participants, video equipment records audio and video of the users behavior together with the screen of the running application. It is best to not at all interfere with the test participants and to isolate the test subjects so that they cannot interfere with each other before, during, and after the test. Of course the tasks should be chosen wisely and probably in order of increasing difficulty. The tasks should be formulated general enough so that a user does not only have follow instructions but has to figure out the steps to achieve the task on his or her own. It may sometimes make sense to ask for tasks that are not achievable. In any ways, to avoid frustration, clear instructions must be given

what to do in the case that a user does not find a solution, eg. that he is allowed to skip a task. One of the most important suggestions with any kind of user study is that the higher the number of participants ("higher n") the better. More participants will not only allow for a significant result but also help to find errors both in the application and in the test setup. Unfortunately, still too many user studies are often performed using "*n=20* students from my department" because a video surveillance test with a high number of users is expensive and time consuming.

For example, a well-known beverage manufacturer created a new vending machine and used a video surveillance test to find out about users' reactions to interacting with the machine. After watching all the videos and discussing between management and developers, they found that everything seemed to be in order, except the user experience can be improved by an option to not only pay with coins but also with bills. So this was added to the machine.

Questionnaires

To reach a greater audience, eg. over the Internet, questionnaires can be used. It is usually best to try a video surveillance test using a small set of participants first, address the issues raised as a result of the test in the application, and then create a questionnaire that targets the improvements in the application along with other issues raised in the video surveillance test. Questionnaires can only get feedback on potential problems that the maker of the questionnaire is already aware of. While adding free form space to a questionnaire is possible, one should not expect the most extensive feedback from it -- after all, testers are not software developers!

Continuing the beverage-manufacturer example: to quickly prove that allowing users to pay with bills was successful, another user study was performed. This time, many more people were asked to participate and the only thing they had to do was fill out a questionnaire rating their experience using the machine on a scale from 1 to 10. As control, the old vending machine was tested with the same questionnaire. Now, most questionnaires for the new machine had a worse score than the control test using the old machine. Development and management were devastated: How could adding a feature that people wanted to a vending machine that was already quite good make the experience so much worse? The solution is this: When the vending machine only accepted coins, participants had been told in advance to bring coins. So the vending went smooth and comments in the videos were mostly about details. Now, that the vending machine also accepted bills, the participants were not told anything about payment modalities. As a result, most participants ended up not having coins and tried the bill slot. This would have been no issue, except the bill slot happened to not have worked very well because the bills could only be inserted in one particular way which the developers had already trained themselves to. As a re-

sult, the machine did not work at all for many subjects since the bill was rejected several times, resulting in many participants just giving up. So while the rest of the vending machine stayed the same, adding a feature made the user experience of the vending machine much worse and finding out about it was hard from the survey questions as developers expected them to be mostly positive.

Good design and thorough reviewing of questionnaires is very important. A large number of well-filled-out questionnaires will lead to significant results that are reportable in scientific publications and to an improvement of the application.

Significance

Especially when surveys are performed for scientific publications the question often arrises whether the outcome of the responses to a question on a questionnaire is statistically significant, i.e. when performed for a second, third, and further time, would the questionnaire lead to the same results? In other words: Is the outcome of this questionnaire random or a valid result. Fortunately, statistics can help here: Significance testing allows the survey analyst to assess evidence in favor of some claim about the participants from which the sample has been drawn.

In the end though, mathematical significance testing is not a guarantee for actual results. Consider the following real-world example: To test a new audio codec, the developers compressed several audio files using the different codecs and let users in the Internet listen to the codecs and rate them according to their perceived quality. To their big surprise, the codec with the highest bandwidth got the worse results, which was obviously better then the worse codec whenever it was presented to individuals outside the test. The test was definitely repeatable and statistically significant. What had happened? The developers had used different audio recordings. However, the one using the highest-bandwidth codec was a speech recording from a non-native speaker which some of the participants had a hard time understanding. This lead the test subjects to give the codec a bad score since they had only been asked to rate “the perceived quality”. So the mistake here was to not eliminate all factors. A short interview after the test might have helped eliminating this oversight in the first place.

Again, this example shows, there is no silver bullet and human-computer interaction is a hard soft topic (hard in the sense of difficult and soft in the sense of not strictly logical). The reader is encouraged to delve into the literature for more information but, most importantly, to go ahead and try it out!

Index Terms

Literature

- Jeff Johnson: GUI Bloopers: *Don'ts and Do's for Software Developers and Web Designers* (Interactive Technologies), Morgan Kaufman, 1st Edition, March 31st, 2000
- Jakob Nielsen: *Usability Engineering*. Academic Press, Boston 1993
- Donald A. Norman: *The Psychology of Everyday Things*. Basic Books, New York 1988
- Jef Raskin: *The humane interface. New directions for designing interactive systems*. Addison-Wesley, Boston 2000.
- Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale (2003): *Human–Computer Interaction*. 3rd Edition. Prentice Hall, 2003.
- Helen Sharp, Yvonne Rogers & Jenny Preece: *Interaction Design: Beyond Human–Computer Interaction, 2nd ed. John Wiley & Sons Ltd., 2007*
- Matt Jones and Gary Marsden (2006). *Mobile Interaction Design*, John Wiley and Sons Ltd.

Web Links

http://www.gui-bloopers.com/

Exercises

1. List examples of thinking inside out that do not pertain to UI development.
2. What are the major advantages and disadvantages of command-line interfaces?
3. Give one example where the desktop metaphor fails and explain why.
4. Create a program to measure the time it takes to find a random specific point on the screen and click on it with the mouse. Then create a program that asks to press for a certain key on a random point on the screen and measure the response time. Compare the two and discuss.
5. Design a task-oriented remote control that can control a TV and a DVR.
6. Explain the notion: “Responsiveness = Perceived Performance” and give more examples of techniques that help increase perceived performance.
7. Find 3 GUI bloopers in the Internet that fall under the category “don’t be creative when you are not supposed to”.
8. Choose an office program (e.g. OpenOffice writer) and create three tasks that a user should solve using the program. Choose three test subjects from your friends and family and watch them while they are performing the task. If you can, you should use a camera to make sure not to interact with them.
9. Repeat 7 but this time create a questionnaire. What is different?
10. If you are restricted in the number of subjects to be participants in a user study (e.g. n=20) -- what is a good strategy to still obtain significant results?
11. Take the clairvoyant example from the text: How many cards have to be tested minimally for the test to be equivalent to a full 25 card test?

Chapter 12: A Note on Privacy and Security Issues

In this chapter, we will discuss some issues that are not so much of technical nature but are a side effect of the power of multimedia. Since multimedia data is directly encoded for human perception and does not necessarily undergo a creator's mind filter (like text), there are some problems that do not usually arise with the creation, distribution, and consumption of other types of data. We think it is of outmost importance for researcher and practitioners in multimedia computing field to always be aware of security and privacy concerns.

Issues of the Effect of Multimedia Data

Probably the most commonly discussed issue of multimedia data is that of the effect of multimedia content on people. Technically, humans could be interpreted as sensors perceiving the data encoded and then interpreting it. This can lead to emotional reactions ranging from entertainment and excitement to sadness and even trauma. The reason for this is that multimedia data allows us to perceive people, objects, events, actions, and places that we would not be able to perceive without multimedia recording.

Naturally, this is foremost a problem in children. Therefore, magazines, movies and computer games are usually rated for a certain age group. While this is true for books as well, it's not as strongly enforced in most places -- another indication for the power of direct encoding for perception. News broadcast usually do not show scenes that are too cruel in order to not expose the audience to material that they may have never experienced and that potentially would lead to trauma and other negative psychologic effects. Some material, such as pornographic videos, is directly targeted to evoke certain emotions and anatomic reactions.

The creation and distribution of certain multimedia material is forbidden in almost all countries. While different countries have different rules and reasons to prohibit handling different types of multimedia data, the cause usually is the same: Society fears the power of multimedia data to directly manipulate people and is still researching long-term effects.

Multimedia data can also cause physical harm. The easiest example is audio that is too loud or too highly pitched. Ear damage is caused by people listening to ear phones too loudly. While this seems almost obvious, it is very important that any multimedia system has to have the ability to be shut down quickly, e.g. with a pause button. When presenting sound, a user friendly method of regulating the volume should definitely provided. People should not be exposed to loud pop sounds and/or be surprised by a sudden increase in volume. Pitch also plays a role. Certain tones can be perceived as uncomfortable when the audience is exposed to them for too long. Also,

there is a myth that low-frequency noise (around 10 Hz) can cause nausia in people (see references). While we could not find literature to validate the myth, this at least shows that people intuitively are not sure about the exposure to certain sounds.

Visual data is also able to do harm. Spending too much time concentrating on low-contrast content, e.g. reading yellow font on white background, will usually be reported as uncomfortable and is often attributed to be a cause for headaches. Stroboscopic light, for example induced by flashing monitors, is reported to cause epileptic seizures in some persons. Many computer games warn about this issue. For the same reason, the Web Content Accessibility Guidelines (WCAG) Version 2.0, produced in 2008, specifies that content should not flash more than 3 times in any 1 second period.

3D video and augmented reality applications are sometimes the cause for sea sickness symptoms. The reason for this is that our visual perception does not exactly match our proprioception and other sensors and the reaction varies from person to person: From no reaction to dizziness and nausia. The effects of 3D displays on young children are currently discussed: Since visual perception may not have fully developed yet, consuming artificial 3D data may cause perceptual issues due to the brain adapting to wrong realities.

Consuming multimedia data can also cause unwanted and potentially life-threatening distractions, especially when operating machinery or driving a car. While it is generally acknowledged that listening to music on moderate levels while driving is not harmful, listing to music using earphones is prohibited in many countries as the music would mask acoustic events from the outside and therefore reduce the ability of the driver to react to them. Watching a movie while driving would make it nearly impossible to drive a car, as the split attention effect (see chapter XXX) would slow down reactions to traffic to a very dangerous level even at low speeds. Operating a cell phone or a navigation system while driving has a similar negative effect.

It is possible to manipulate a human being in a very subtle way and potentially cause deep emotions (wether planned or not). Therefore it is very important for multimedia researchers to design systems in a way that it obeys the laws of different countries when handling multimedia data globally and make sure to prevent any harm from the people exposing the data to.

Privacy Issues

Multimedia data, especially acoustic and visual data, but in general data that is directly encoded for perception has the property of always conveying more information than is intended. The reason for it is the information richness: In contrast to written text, multimedia captures a snapshot

of reality unfiltered. Even in staged scenarios, such as movies, people find bloopers that were not intended to be shown and were not part of any script. Even more so, spontaneous recordings such as vacation photographs of home videos always show lots of information that was not in the focus of the attention of the creator but is still there. This must be taken into account when publishing photographs, audio, and video recordings on the Internet or elsewhere.

Especially the wide spread use of social Internet sites to dissiminate multimedia materials has been questioned as problematic. First, many recordings may contain people and.or objects that require permission to be published. In many countries publishing a face photograph of a third person requires permission of that person. Copyrighted objects, such as a painting in the background, require permission of the author or current copyright holder.

Also, both human intelligence as well as multimedia retrieval may be used on multimedia data to extract some of the unintended information. For example, face recognition or speaker identification (see Chapters XXX) may be used to find the identity of a person, even if the website has no other indication for a person's identity. This could potentially de-anonymize website postings and accounts. For example, a person having videos and photos on an otherwise anonymous might be identified by the picture or voice by people who know them. Furthermore, multimedia retrieval, even though not perfectly accurate, makes it unsafe to say that data would be anonymous on the site: Only one match of a face or voice between the anonymized site and a public site with identity is enough to expose the user.

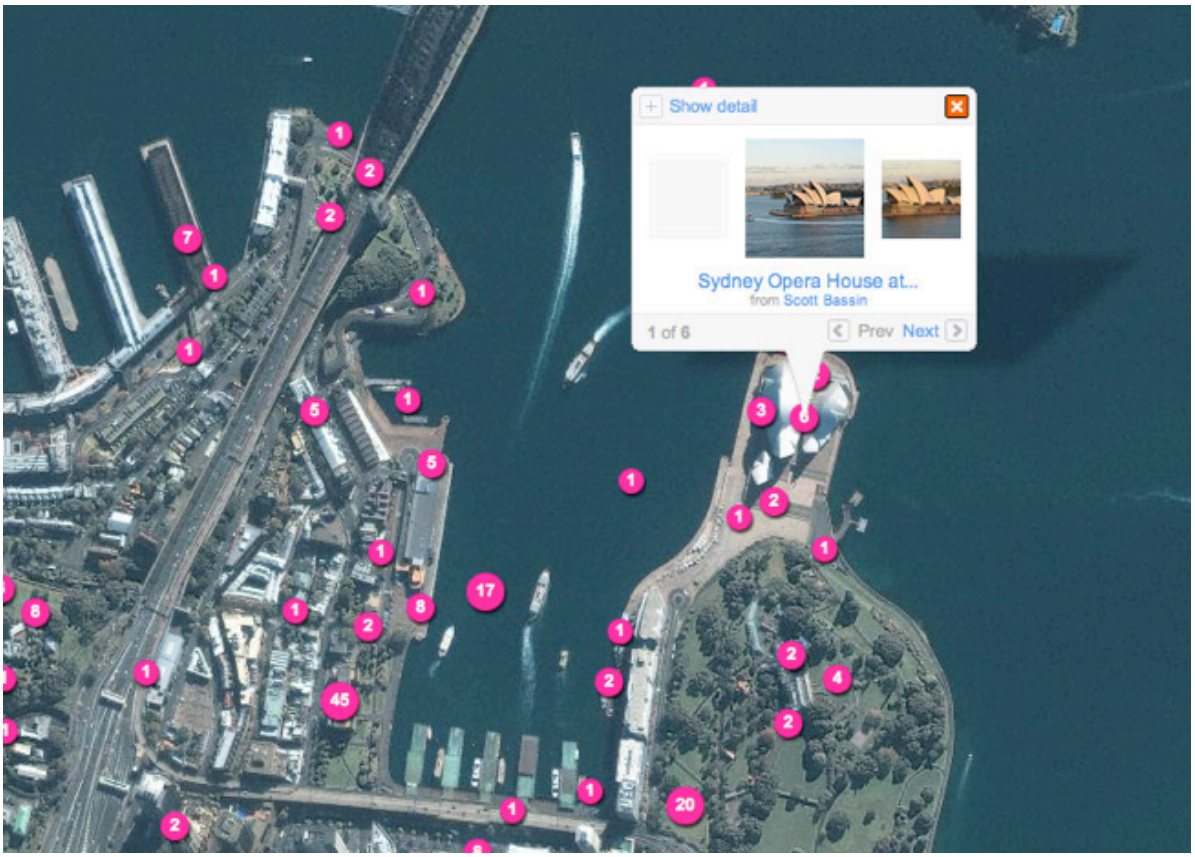


Figure 1. An example of geo-tagging allowing for easier search and retrieval of images. The photo shows a partial screenshot of flickr.com.

Even when such subtle methods like the ones described above are not in question, metadata and annotation might become an issue too. For example, tagging somebody publicly in a picture makes searching the person much easier as textual search has higher accuracy than multimedia search. However, this implies that a photo of a person has become public without the person potentially not knowing about it. This can be especially problematic because the person in question might have opted to not participate in the social networking site and is now forced to do so. Often, cameras and multimedia editing tools embed metadata in videos and images. The most common being the so-called EXIF header in JPEG images. EXIF can contain detailed information about the camera, including serial numbers, which potentially makes the creator of the photo trackable. Most importantly, EXIF data can contain geo-tags, i.e. longitude and latitude coordinates of the place where the photo has been taken. Figure 1 shows an example. This is enabled through different localization systems, such as GPS, cell-tower information, as well as wireless network SSID-maps. Geo-tags in combination with time allow a reasonable easy tracking of a person. Also, implicitly exposing places, such as home addresses, can be potentially dangerous

as it allows a variety of crime scenarios. Often, taking a photograph of an object is a matter of seconds and the creator is often not thinking about metadata, such as GPS tags. As a result, valuable items have been posted on the web including a complete address, whilst the post was anonymized otherwise (see references).

Since a potential attacker might gather information from different websites and infer from the collective, it becomes even harder to track what information is out there about an individual or entity. So conserving privacy becomes a difficult task, once a considerable amount of information has been published.

Time is another variable in the equation. What people might find adequate to post and publish now, might later be an issue for their reputation. The need for privacy shifts with age, social status, occupation, and other factors. On the other hand, the Internet potentially saves material forever and in many copies. Once data has been published, it might be downloaded by various search engines for caching and individuals might be wanting to re-post it. Seemingly innocuous information might later become embarrassing whether posted intentionally or unintentionally.

As multimedia researchers, we have special responsibility in this regard as we are the original enables of most of the technology. Many people are unaware that multimedia retrieval technologies exist and even many experts often fail to assess the capabilities of today's multimedia retrieval technologies and the potential inference possibilities. Even though current retrieval methods are not perfectly accurate, with billions of pictures and millions of videos available, even a seemingly small fraction of true positive results can already translate into several hundred relevant matches. Those matches might enable de-anonymization, finding potential victims for crimes, or reveal other secrets.

Countermeasures

The first thing to do is checking whether a particular post is really necessary. What is the target group of the post and is it possible to make the post only available to that group.

Then, the most important privacy-conserving counter measure is to make sure only information is published that is supposed to be published. When publishing photos, for example, these should not contain humans that were not involved and if so, faces and other identifying characteristics should be blurred. Likewise, multimedia data should be checked for hidden metadata. If metadata is to be included, the level of detail should be controllable. For example, serial numbers of the camera might not be actually needed and geo-tags might be included with a reduced accuracy that is enough to organize the data but not enough to pinpoint individual addresses. Inserting

noise into the signal which might or might not be audible might help to make it more difficult to apply acoustic matching. Applying compression often helps to reduce camera artifacts.

The above are only a few examples of both privacy issues and countermeasures. Spotting the issues and inventing new methods to prevent them is an active area of research and the reader is therefore suggested to check out the literature.

Security Issues

Apart from what has been said about the possible effects of the consumption of multimedia data in humans, multimedia data has also often attributed to computer security issues. Malware has been embedded in various forms into videos and presentations. This is mostly enabled through the fact that some multimedia data formats are turing complete, i.e. their representation is powerful enough to allow the execution of arbitrary programs. Prominent examples of these formats are Postscript and Flash, the first being a printer language, the second being an interactive video and animation format. PDF, that replaces postscript, is therefore not a Turing complete programming language anymore.

Security breaches have also been reported, exploiting buffer overruns in multimedia formats. Often, decoders and viewers of multimedia data are tailored to and tested only with regular images and videos. However, the nature of compression formats makes it possible to artificially create a video or image file that expands into a super large file upon decompression. The memory used to hold that overlarge data chunk potentially overwrites code, which then leads to crashes, or when done carefully could be used to insert malicious code into the decoder/viewer. Several viruses have been reported doing that. These viruses are especially powerful because they spread when a person views an image or video embedded in a webpage or email, thereby not even consciously executing a program.

Reportedly, there have been programs using microphones and cameras built into laptops without notifying the user. This spy-behavior inadvertently leads to trust issues as the user should be always informed about the fact that he or she is being recorded. More so, it should be clear what is being done with the recording and how to stop the recording at any time. A video feedback helps improve a user's trust in what's being recorded and or transmitted. Needless to say, in many countries recording a person without their knowledge is against the law.

Again, this chapter only provides an introduction to the most pressing topics in the field. Privacy and security issues are hard to convey comprehensively a book as they are usually a natural side effect of a new technology. However, whenever creating a new technology we ought to ask our-

selves about the impact on privacy and any potential security issues that may raise. Everybody working in the field should develop a common sense for these issues and not hesitate to question methods that might have unwanted side effects.

Index Terms

Literature

- Durham, M. & Kellner, D. (2001), *Media and Cultural Studies*. UK: Blackwell Publishing
- *Fowles, Jib (1999), The Case for Television Violence*, Thousand Oaks: Sage
- *Gauntlett, David (2005), Moving Experiences - Second Edition: Media Effects and Beyond*, London: John Libbey
- Anderson, C. A. & Bushman, B. J. (2001) Media Violence and the American Public: Scientific Facts Versus Media Misinformation. *American Psychologist*
- Prinz, W. (1990). A common coding approach to perception and action. In O. Neumann and W. Prinz (Eds.) Relations between perception and action. Berlin: Springer.
- Mussen, P., & Rutherford, E. (1961). "Effects of aggressive cartoons on children's aggressive play" *Journal of Abnormal and Social Psychology*, 62, 461-464. PubMed
- Jones, Gerard *(2002). Killing monsters: why children need fantasy, super heroes and make-believe violence*. New York: Basic Books ISBN 978-0465036967
- Bureau M, Hirsch E, Vigevano F (2004). "Epilepsy and videogames". *Epilepsia* 45 Suppl 1: 24–6. PMID 14706041
- Web Content Accessibility Guidelines (WCAG) 2.0 - W3C Recommendation 11 December 2008: http://www.w3.org/TR/2008/REC-WCAG20-20081211/
- CERT: Email bombing and Spamming: http://www.cert.org/tech_tips/email_bombing_spamming.html
- Harold, Elliotte Rusty (27 May 2005). "Tip: Configure SAX parsers for secure processing". *IBM developerWorks*.

Research Articles

- So, R.H.Y. and Lo, W.T. (1999) "Cybersickness: An Experimental Study to Isolate the Effects of Rotational Scene Oscillations." Proceedings of IEEE Virtual Reality '99 Conference, March 13–17, 1999, Houston, Texas. Published by IEEE Computer Society, pp.237-241.
- G. Friedland, R. Sommer: Cybercasing the Joint: On the Privacy Implications of Geotagging, accepted for Usenix HotSec 2010 at the Usenix Security Conference, Washington DC, August 2010.

Exercises

1. Find examples in the media where multimedia data is attributed to a crime. Discuss the coverage with your fellow students.

2. Enter your name into a search engine. Count the number of occurences where a) the content is about you and b) the content is about you but not published by yourself. Find the oldest post about you.
3. List potential multimedia retrieval technologies (from this book and elsewhere) that could be used to invade privacy. Discuss possible counter measures.
4. Describe an inference chain over several websites that would compromise privacy. Discuss a second one that includes multimedia data.
5. Find your personal privacy sweet spot by discussing what would still be OK to be published about you and what would not.
6. Discuss the buffer overrun mentioned in the chapter in detail by taking one of the entropy compression algorithms from chapter XXX and creating a file that would expand into a very large file.

PART IV:
COMPRESSION

Chapter 13: Fundamentals of Compression

A major difference between multimedia data and most other data is its size. Images and audio files take much more space than text for example. Video data is currently the single largest network bandwidth and hard disk space consumer. One of the earliest topics in multimedia was therefore compression. In fact, multimedia's history is closely connected to different compression algorithms because they served as enabling technologies for many applications. Even today, multimedia signal processing would not be possible without compression methods. A Blue Ray disk can currently store 50 Gbytes, but a 90-minute movie in 1080p HDTV format takes about 800 Gbytes (without audio). So how does it fit on the disk?

This chapter discusses compression's underlying mathematical principles, from the basics to advanced techniques.

Run-Length Coding

Before discussing what compression is and how you can develop algorithms to represent different types of content with the least amount of space possible, let's start with a simple and intuitive example. In addition to introducing what compression is, this example demonstrates a practical method that computer scientists often use to compress data with large areas of the same values.

Suppose we have a black and white image encoding the character 0. The black pixels are encoded with 1 and the white pixels are encoded with 0. So, it looks like this:

```
0000000000000000
0000011111100000
0000100000110000
0000100011010000
0000100110010000
0000111000010000
0000011111100000
0000000000000000
```

As is, the bitmap representation would take $16 \times 8 = 128$ zeros and ones. Say we want to cut the number of zeros and ones. There are several ways to do it.

First, observe that there are many zeros and ones in a row. We could represent these characters only by the number of consecutive zeros and ones starting with the zeros. The result would be:

```
21 6 9 1 5 2 8 1 3 2 1 1 8 1 2 2 2 1 8 6 21
```

The highest number represented is 21. So, we could represent each of the numbers in the above row with 5 bits: 101010011001001... and so on. We need 21 numbers in this encoding, so we can represent the digit in 105 bits, for a total savings of 23 bits. Of course, we would have to represent a bitmap containing more than two colors—say, 16—slightly differently. Consider the following one-line example:

```
RRRRRGGGBBBBBRRRRGB
```

Using a variant of the above described concept of representation, we would get:

```
5R3G5B4R1G1B
```

This method—called *run-length encoding*, or RLE—is used in many image formats, such as Windows BMP and TIFF, and, in a slightly modified version, on CDROMs because it is especially effective in representing data with large areas of the same values. Several unanswered questions remain, however. For example:

- Is RLE the best way to compress the above example?
- Would applying RLE to the file again further compress the bitmap?
- Finally, what is the minimum amount of space needed to represent this image?

Answering these questions requires a more in-depth discussion of the topic. In fact, it leads to an entire theory in mathematics, the so-called information theory. The following sections introduce the most important concepts in information theory.

Information Content and Entropy

To determine how far we can compress a certain piece of data, we first need a measurement for information. The smallest amount of information is the bit. A bit is a symbol that can be 0 or 1. Every string in the world can be reduced to bits, so we could say that one measure for information is the minimum number of bits needed to represent a string. But how can we calculate this number?

Assume we have a 64-bit-long string consisting of 63 zeros and 1 one. The one can be at any place in the string. We denote this place with an i, so i is a number between 0 and 63. Next, assume we read the string from the left to the right, symbol by symbol, until we find the one. We can then ask: At each character, what is the probability P that the next character will be a one or a

zero, given that we have not yet found the one? Table 1 shows the probability P for each bit, from 1 to 64.

i	1	2	3	4	....	32	33	...	62	63	64
$P(b_i = 0)$	63/64	62/63	61/62	60/61		32/33	31/32		2/3	1/2	0
$P(b_i = 1)$	1/64	1/63	1/62	1/61		1/33	1/32		1/3	1/2	1

Table 1: The probability that a bit (with the symbol 0 or 1) will appear in a particular place on string (*b* = the bit, *i* = the bit's place on a string, and P = probability).

As the table demonstrates, the probability P depends only on the index until the one is found. Of course, the probability after reading 63 zeros is 1 and the probability after reading the one is 0. In other words, the zeros after a one are completely predictable and carry no information. We can also reverse this argument: The higher an event's probability, the less information it carries. This means the *information content* is inversely proportional to the probability of a symbol's occurrence. What's left is to count how many characters we actually need to represent the content. In the binary system, this is the number of digits needed. We therefore define the information content $h(x)$ as:

$$h(x) = \log_2 \frac{1}{P(x)} \quad (1)$$

The choice of a logarithmic base corresponds to the choice of a unit for measuring information. If we use base 2, the resulting units are bits (as in this example). If we use another alphabet (for example, the decimal system), we must adjust the base accordingly.

So, let's measure the information content for each symbol for the given example by inserting the values from Table 1 into the formula in Equation 1. Table 2 shows the results.

i	1	2	3	4	....	32	33	...	62	63	64
$h(b_i = 0)$	0.0227	0.023	0.0235	0.0238		0.444	0.458		0.585	1	0
$h(b_i = 1)$	6	5.977	5.954	5.9307		5	5.0666		1.585	1	0

Table 2: The information content of each symbol calculated from the probabilities in Table 1.

The sum of the information content of bits 1 to n is 6 when the n-th bit is a one. For example, $h(00010\ldots0) = 0.0227 + 0.023 + 0.0235 + 5.9307 + 0 + \ldots + 0 = 6$

So, in general, the formula is:

$$\sum_{n=1}^{i} h(b_n) = \left(\sum_{m=66-i}^{64} \log_2 \frac{m}{m-1}\right) + \log_2 \frac{1}{1/(65-i)} = \log_2\left(\left(\prod_{m=66-i}^{64} \frac{m}{m-1}\right) * (65-i)\right) = 6 \quad (2)$$

The summation of the information content for each symbol reveals the string's total information content. In other words: We need a minimum of six bits to represent the string as described. To do this, all we need to save is the index i (a number between 0 and 63), which can be represented by a six-digit binary number.

There is a caveat, however. To measure the string's information content, we use probabilities, which requires knowledge about the data structure. In other words, if we don't know how many ones appear in the string and the string's total length, we can't calculate the information content. Again, this is intuitive. For a string to contain any information, someone or something must process and interpret the string. That is, a string must be put into a context to make sense.

Information content gives us a method to measure the number of bits required to encode a certain message. In practice, however, the message might not be known in advance. Instead, you might have a language and arbitrary messages encoded in that language. Formally, we can define a language as a set of symbols or words. Each word or symbol has an associated probability. The probability can usually be determined by the frequency of a symbol or word appearing in the language. In English, for example, the word "the" appears more often than the word "serendipity." As a result, the probability of an article ("the") appearing in a message is much higher than that of the word derived from the old Persian name for Sri Lanka ("serendipity's" origin).

So, what is the expected length of a message given a set of symbols and their probabilities? This measure—*entropy*—is defined as follows:

$$H(X) = \sum_i P(x_i) * h(x_i) \quad (3)$$

Entropy is the expected information content—not coincidentally similar to probability theory's expectation value. Entropy is an important measure used across scientific disciplines. Shannon's source coding theorem (Shannon 1948) gives the information theoretic background for entropy. The theorem shows that entropy defines a lower bound for the number of bits that can be used to encode a message in that alphabet without losing information. Entropy's value is maximized

when the alphabet is uniformly distributed—that is, each character has the same probability. This is also often referred as chaos: Given a subset of a string, there is no possibility of a prediction about the next symbols because all symbols have equal probability. Random noise has this property and is therefore incompressible.

Entropy is a fundamental measure with analogies in thermodynamics, quantum mechanics, and other fields. In general, it can be described as a measure of chaos: The more chaotic a system, the higher its entropy. In information theory, this means that the more chaotic a string is, the more bits we need to encode it.

Compression Algorithms

Information content and entropy lets us measure a message's expected minimum length. However, given a string, how do you construct a code for that string that uses the minimal number of bits? Unfortunately, no single answer to this problem exists. There are many ways to construct the string and none of them achieves perfect results in all cases. Therefore, the best compression method depends on the data you are processing. The following sections present the most important compression algorithms.

Huffman Coding

Before we dig into the details of how to create good compression algorithms, let's quickly review the features we would expect from a good encoding:

- The code must be unambiguously decodable—that is, one coded message corresponds to exactly one decoded message.
- The code should be easily decodable—that is, you should be able to find symbol endings and the end of the message easily. Ideally, you should be able to decode it online (that is, as the symbols come in) without having to know the entire coded message.
- The code should be compact, only delimited by entropy.

In 1952, David A. Huffman invented a coding algorithm that produces a code fulfilling these requirements. *Huffman coding* is in frequent practical use and part of many standard formats. The algorithm's design is elegant, easy to implement, and efficient. Also, because of the vast mathematical work that's been done on tree structures, the algorithm is well understood.

The algorithm constructs a binary tree in which the symbols are the leaves. The path from the root to a leave reveals the code for the symbol. When turning right on a node, a 1 is added to the code; when turning left, a 0 is added. The idea is to have long paths for symbols that occur infrequently and short paths for symbols that occur frequently.

Figure 1 illustrates the construction of a Huffman tree. In the beginning, the leave nodes containing the frequencies of the symbol they represent are unconnected. The algorithm creates a new node whose children are the two nodes with the smallest probabilities, such that the new node's probability equals the sum of the children's probability. The new node is treated as a regular symbol node. The procedure is repeated until only one node remains—the root of the Huffman tree.

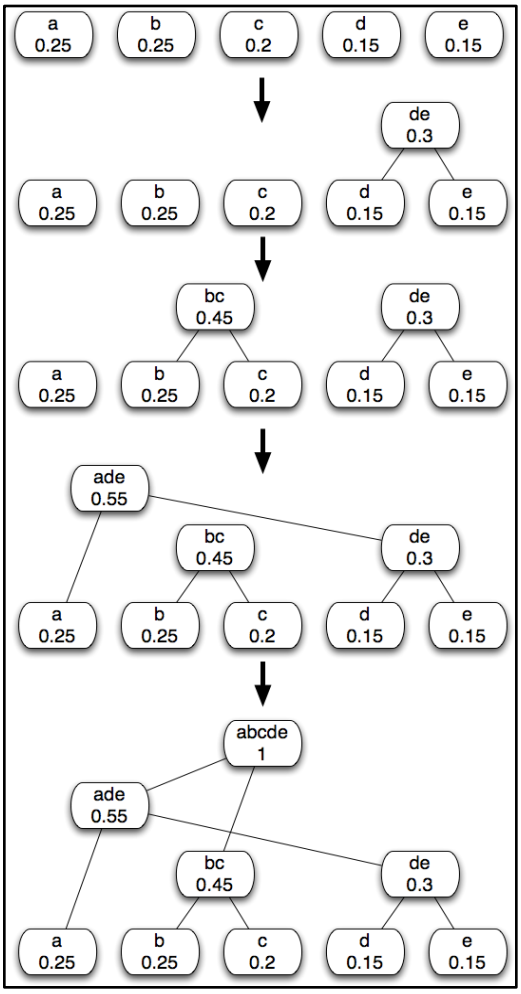


Figure 1: Construction of a Huffman tree.

The simplest construction algorithm uses a priority queue in which the node with the lowest probability receives highest priority. The following pseudo code illustrates the process:

```
// Input: A list W of n frequencies for the symbols
// Output: A binary tree T with weights taken from W
Huffman(W, n):
    Create list F with single-element trees formed from elements of W
    WHILE (F has more than one element)
```

```
            Find T1, T2 in F with minimum root values
            Create new tree T by creating a new node with root value T1+T2
            Make T1 and T2 children of T
            Add T to F
  Huffman := tree stored in F
```

You can now use the tree to encode and decode any message containing symbols of the leave nodes, as Figure 2 shows.

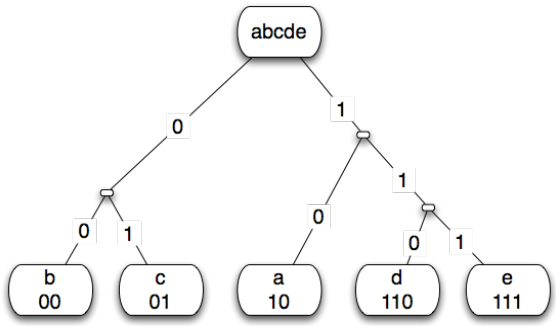


Figure 2: Codes generated by the Huffman tree

To encode a message, the algorithm traverses the tree completely and saves the paths into a lookup table. So, in our example, the encoding for the word "cab" is 011000. The following pseudo code describes the procedure:

```
// Input: A Huffman tree F and a string S containing only symbols in F
  // Output: A bit-sequence B
Encode_Huffman(F, S)
    Traverse F completely and create a hash table H containing tuples (c,p)
    // c = character, p = path from root to of F to character
    B := []
    FOREACH (symbol s in S)
          Add path H(s) to bit sequence B.
Encode_Huffman := B
```

To decode a message, the algorithm interprets the code as encoding a path from the root to a leaf node, where 0 means going right, and 1 means going left (or vice versa), as described in the following pseudo-code sequence:

```
// Input: A Huffman tree F and a bit sequence B
  // Output: An uncompressed string S
Decode_Huffman(F, B)
    Traverse F completely and create a hash table H containing tuples (p,c)
    // c = character, p = path from root to of F to character
    S := ""
    node n = root node of F
    FOREACH (bit b in B)
          if (b==0) n := left children of n
          if (b==1) n := right children of n
```

```
            if (n has no children)
                  Add character represented by n to S
                  n := root node of F
  Decode_Huffman := S
```

Static Huffman compressors use a fixed tree for all incoming data; dynamic Huffman compressors serialize the tree, so that each file can be decoded using a different tree. The code generated by this algorithm is optimal in the sense that each symbol has the shortest possible representation for the given probability. However, the Huffman algorithm assumes that symbol frequencies are independent of each other. In reality, a "q," for example, is usually followed by a "u." In addition, the Huffman algorithm cannot create codes with a fraction of bits per symbol.

Lempel-Ziv Algorithms

The LZ family of algorithms (derived from their creators'—Abraham Lempel and Jacob Ziv—last names) is divided into two groups: successors of the LZ77 and successors of the LZ78 algorithm. Lempel and Ziv developed the base algorithms in 1977 and 1978, respectively. The algorithms use completely different approaches, despite their similar names. Most variations of the LZ algorithms can be identified by the third letter—for example, LZH or LZW. In contrast to Huffman coding, which is based on symbol probabilities, the algorithms account for repeated symbol combinations (aka words) in a document. The algorithm is best suited for large archives containing text and/or source code and two very popular image formats, GIF and PNG, use one variant of each of the algorithm. The LZ compression algorithms are therefore the single most often used compression algorithms of all times. Once one has understood both LZ77 and LZ78, the derivate algorithms are conceptually not very different. This section will therefore describe the two fundamental algorithms: LZ77 and LZ78.

LZ77 achieves compression by replacing portions of the data with references to matching data that have already passed through the encoder and decoder. This algorithm works with a fixed number of symbols—the *look-ahead buffer* (LAB)—that are to be coded. Additionally, the algorithm looks at a fixed number of symbols from the past—the *search buffer* (SB). To encode the symbols in the LAB, the algorithm searches the SB backward for the best match. The encoding (often called a *token*) is encoded by a tuple (L,D) defining the length and the distance from the current token to a past token. A token basically says, "each of the next L symbols is equal to the D symbols behind it in the uncompressed stream." In actual implementations, the distance D is often referred to as *offset*. The current chunk of processed symbols—that is, SB+LA—is often called a *sliding window*. Typical sizes for the sliding window are several kilobytes (2 kB, 4 kB, 32 kB, and so on).

Consider the example in Table 3. It encodes the string “cabbdcbaacbdddabda” using LZ77. The currently processed symbol is underlined.

		Sliding window			
Position	**Code to be processed**	**Look-ahead buffer (next three symbols)**	**Search buffer (previous seven symbols)**	**Already encoded**	**Token**
	cabbdcbaacbdddabda				
1	bdcbaacbdddabda	cab			(0,0,c)
2	dcbaacbdddabda	abb	c		(0,0,a)
3	cbaacbdddabda	bbd	ca		(0,0,b)
4	baacbdddabda	bdc	cab		(1,1,d)
6	acbdddabda	cba	cabbd		(5,1,b)
8	bdddabda	aac	cabbdcb		(6,1,a)
10	ddabda	cbd	bbdcbaa	ca	(4,2,d)
13	bda	dda	cbaacbd	cabbd	(1,2,a)
16		bda	acbddda	cabbdcba	(5,2,a)
19			dddabda	cabbdcbaacb	

Table 3: Encoding of the string “cabbdcbaacbdddabda” using the LZ77 compression algorithm.

The LAB is also used as the SB, as position 13 demonstrates. The original LZ77 algorithm uses a fixed-size SB so can use constant-bit-length tokens. The encoding is the final token. For our example, the encoding is 0,0c 0,0a 0,0b 1,1d 5,1b 6,1a 4,2d, 1,2a 5,2a

The following is the pseudo-code for an LZ-77 encoder:

```
// Input: A lookaheadbuffer LAB a substring of a message...
  // Output: An LZ77 encoding C
Encode_LZ77(LAB)
    C := ""
    WHILE (LAB not empty)
        p := position of the longest match in the window for the LAB
        l := length of the longest match
        Add the tuple (p, l) to C
        Add the first character in LAB to C
        Shift LAB by l
Encode_LZ77 := C
```

To decompress the code, the algorithm reads the constant-sized tokens. The distance and length always refer to already decoded values. Choose a string and try this algorithm for yourself . You may use the following pseudo code as a guidance:

```
// Input: A string C containing LZ77 code
  // Output: A string S containing the decoded code
Decode_LZ77(C)
    S := ""
    FOREACH (token in S)
        Read token and obtain the triple (p,l,c)
        // p = position, l=length, c=character
        Add to C the l characters from position length(C)-p
        Add to C the character c
Decode_LZ77 := S
```

As explained earlier, many variations of the LZ77 algorithm exist. LZR, for example, has an unrestricted search buffer and therefore variable-bit-length tokens. In LZH, a commonly used variant, the token values are compressed using a Huffman encoding. The currently most popular LZ77-based compression method is called DEFLATE, which is for example part of the very common Unix compression program "gzip".

DEFLATE combines LZ77 with Huffman coding, placing literals, lengths, and a symbol to indicate the end of the current block of data together in one alphabet. It places distances into a separate alphabet. The image format PNG (Portable Network Graphics, see also Chapter XXX), also uses a variation of DEFLATE. PNG combines DEFLATE with a filter to predict each pixel's value based on previous pixels' colors, subtracts the prediction from the actual value. Both encoder and decoder use the same prediction table. This way, only the prediction differences are transmitted. An image line filtered this way is usually more compressible than the raw image line because DEFLATE does not understand that an image is a 2D entity. It sees the image data as a stream of bytes, whereas the prediction accounts for neighboring pixels in all dimensions. We discuss this algorithm in more detail in Chapter XXX.

Whereas the LZ77 algorithm works on past data, the LZ78 algorithm attempts to work on future data. It achieves this by maintaining a dictionary. The input buffer forward-scans the input buffer and matches it against the dictionary. The algorithm scans the input buffer for the longest match of the buffer with a dictionary entry until it can no longer find a match. At this point, it outputs the location of the word in the dictionary (if one is available), the match length, and the character that caused a match failure. It then adds the resulting word to the initially empty dictionary. Table 4 shows an example in which the word "abacbabaccbabbaca" is compressed using LZ78.

Step	Input	Token	New dictionary entry/ Index
1	a	0,a	a,1

2	b	0,b	b,2
3	ac	1,c	ac,3
4	ba	2,a	ba,4
5	bac	4,c	bac,5
6	c	0,c	c,6
7	bab	4,b	bab,7
8	baca	5,a	Baca,8

Table 4: Compressing the string "abacbabaccbabbaca" using the LZ78 algorithm.

Like in LZ77, the tokens are the actual coding. Therefore, the code for our example is "0a0b1c2a4c0c4b5a."

```
// Input: A string S
// Output: A string C containing the LZ78 code
Encode_LZ78(S)
    Start with empty dictionary D
    C := ""
    prefix := "" // stores the prefixes
    FOREACH (character c in S)
        IF (prefix+c is in D)
            prefix := prefix + c
        IF (prefix + c is not in D)
            Add the string prefix+c to D
            Add the index of prefix in D to C
            Add c to C
            prefix := c
Encode_LZ78 := C
```

Decompression of LZ78 tokens is similar to compression. The algorithm extends the dictionary by one entry when it decodes a token using the dictionary index and the explicitly saved symbol.

```
// Input: An LZ78 encoded string C
// Output: A string S containing the original message
Decode_LZ78(C)
    Start with empty dictionary D
    oldindex := indexvalue of the first token in C
    S := ""
    FOREACH (token t in C)
        index := indexvalue of t
        IF (D has entry with index)
            Add the string at index to S
            c := first character of the string at index
            Concatenate entry at oldindex and c and add to D
        IF (D has no entry with index)
            c := first character of the string at oldindex
            Concatenate entry at oldindex and c and add to D
            Concatenate entry oldindex with c and add to S
        oldindex := index
Decode_LZ78 := S
```

Both compression and decompression benefit from an easy-to-manage dictionary. Usually, you would use a tree in which each node has a certain number of children that equals the number of valid input symbols. In the original LZ78 algorithm, the dictionary's size is unrestricted. Therefore, the index values must be saved with a variable number of bits. The dictionary index length is rarely explicitly defined; the algorithm uses dictionary's size to determine the index's bit length. In other words, the algorithm allocates enough bits so that the largest index can be stored. For example, for a 24-bit dictionary, all we need is $[\log_2 24] = 5$ bits per index.

LZ78 has many variants too: LZC uses a maximum size for the dictionary. If it reaches the maximum number of entries, the algorithm continues with the current dictionary under the hope that the output file does not become too long. If it determines that the output length has passed a threshold, it recompresses the data by creating an additional dictionary. The Unix compress tool uses LZC. However, the LZC is patented so users must pay a license fee. LZW, a common LZ78 variant, does not store the following symbol explicitly but as the first symbol of the following token. The dictionary starts with all possible input symbols as first entries. This leads to a more compact code and lets users define the input symbols (which can vary in bit length). The popular Graphics Interchange Format (GIF) uses the LZW variant. Although initially popular, enthusiasm for LZ78 dampened, mostly because parts of it were patent-protected in the United States. The patent for the LZW algorithm was strictly enforced and led to the creation of the earlier-mentioned patent-free PNG image format.

As mentioned earlier, the RLE algorithm is most useful when the same characters repeat often and Huffman compression is most useful when you can build a non-uniformly distributed probability model of the underlying data. The LZ algorithms are especially useful with text-like data—that is, data where strings of limited, but variable lengths repeat themselves. Typical compression ratios are (original:compressed) 2:1 to 5:1 or more for text files. In contrast to RLE and Huffman, LZ-algorithms need a certain input file size to amortize. Compressing a file with just a few bits, such as our example from the beginning of the chapter, won't yield a very good compression result. The Unix program "tar", for example, therefore concatenates all files into one large archive and then invokes "gzip" on the entire archive.

Arithmetic Coding

Arithmetic encoding approaches seek to overcome Huffman encoding's limitations—namely, that messages can only be encoded using an integer number of bits per symbol. JPEG image compression and other standards use arithmetic coding.

Arithmetic coding maps every symbol to a real number in the open interval between 0 and 1, formally $[0,1) \subset R$. Because this interval contains nondenumerable infinite elements, every message can be mapped to one number in this interval. A special termination symbol denotes the end of a message. Without this symbol, you would not know when to stop the decoding process. To encode a message, arithmetic coding approaches partition the start interval [0,1) into subintervals sized proportionally to the individual symbols' probabilities. The algorithm is as follows.

Choose the symbol with the highest probability and split the interval according to its probability. Using the reminder of the interval, repeat the process until you reach the symbol with the lowest probability. Table 5 illustrates the process for the string "bac#" ("#" denoting the end symbol) with the probabilities given in the first row. The table also shows how the actual binary code is created by converting the decimal representation to binary.

x_i	A	B	c	#
$P(x_i)$	0.5	0.2	0.2	0.1
Step 1: read "b"				
Partition (decimal)	$[0,0.5)$	$[0.5,0.7)$	$[0.7,0.9)$	$[0.9,1)$
Partition (binary)	$[0,0.1)_2$	$[0.1,0.1\overline{0110})_2$	$[0.1\overline{0110},0.1\overline{1100})_2$	$[0.1\overline{1100},1)_2$
Step 2: read "a"				
Partition (decimal)	$[0.5,0.6)$	$[0.6,0.64)$	$[0.64,0.68)$	$[0.68,0.7)$
Partition (binary)	$[0.1,0.\overline{1001})_2$	$[0.\overline{1001},0.1010001...)_2$	$[0.1010001...,0,1010111...)_2$	$[0.1010111...,0,1\overline{0110})_2$
Step 3: read "c"				
Partition (decimal)	$[0.5,0.55)$	$[0.55,0.57)$	$[0.57,0.59)$ $[0.100100011...,0.100101110...)_2$	$[0.59,0.6)$
Step 4: read #				
Partition (decimal)	$[0.57,0.58)$	$[0.58,0.584)$	$[0.584,0.588)$	$[0.588,0.59)$ $=[0.100101101...,0.100101110...)_2$

Table 5: Steps involved in encoding the string "bac#" using arithmetic coding.

The message "bac#" is encoded as a number in the interval [0.588,0.59) or [0.100101101 . . . ,0.100101110 . . .) binary. More exactly formulated, this interval contains all messages starting with "bac." The decode, however, stops because it reads the terminal symbol "#." Because the start interval is open and does not contain the one, there is no need to transmit the numbers be-

fore the point. So the code for "bac#" is 10010111. The following pseudo code snippet illustrates the idea:

```
// Input: A string S containing a message, ending with a stop symbol
// Output: An arithmetically encoded string C,
  // a table P mapping probability ranges to symbols
Encode_Arith(S)
     FOREACH (character c in S)
         Get the frequency and assume as probability p[c]
     Create a table P that assigns characters to probability
       ranges in [0,1), each range with a size proportional to p[c]
     lower_bound := 0
     upper_bound := 1
     FOREACH (character c in S)
         current_range := upper_bound-lower_bound
         upper_bound := lower_bound+(current_range*upper_bound[c])
         lower_bound := lower_bound+(current_range*lower bound[c])
     C=upper_bound+lower_bound/2.0
Encode_Arith := (C, P)
```

To decode the message, the algorithm needs the input alphabet and the intermediate partitions. Given an encoded message, the algorithm then chooses the subinterval containing the code in each intermediate partition until the algorithm finds the termination symbol. The table with intermediate partitions is rarely transmitted; instead, it's typically generated identically by the coder and decoder. The following lines of pseudo-code illustrate the decoding:

```
// Input: An arithmetically encoded string C,
  // a table P mapping probability ranges to symbols
// Output: A string S containing the orginal message
Decode_Arith(C, P)
    encoded_value := C
    WHILE (we have not seen the terminal symbol)
            s := symbol in P where encoded_value is within its range
            //remove effects of s from encoded_value
            current_range = upper_bound of c - lower bound of c
            encoded_value = (encoded_value-lower_bound of c)/current_range
           Add s to S
Decode_Arith := S
```

A major problem in implementing arithmetic codes is that the interval boundaries must be accurately represented. Standard processors use single- (32 bit) or double-precision (64 bit) floating-point numbers, but this representation is, of course, not precise enough. Different arithmetic encoders use different tricks to overcome this problem. Rather than try to simulate arbitrary precision (which is a possibility but very slow), most arithmetic coders operate at a fixed limit of precision. The coders round the calculated fractions to their nearest equivalents at that precision. A

renormalization process keeps the finite precision from limiting the total number of symbols that can be encoded. Whenever the range decreases to the point at which the interval's start and end values share certain beginning digits, the coder sends those digits to the output, thus saving the digits in the CPU, where the interval boundaries shift left by the number of saved digits. This lets the algorithm add an infinite number of new digits on the right, even when the CPU can handle only a fixed number of digits.

Various multimedia data formats (such as JPEG) use variants of arithmetic coding. However, US patents cover several specific arithmetic coding techniques. For that reason, encoders and decoders of the JPEG file format typically only support Huffman encoding. In addition, although every arithmetic encoding implementation achieves a different compression ratio, most compression ratios vary insignificantly (typically within 1 percent). However, the CPU time varies greatly—easily an order of magnitude depending on the input. This runtime unpredictability is, of course, a major usability concern and therefore another reason for not choosing arithmetic coding.

Weakness of Entropy-based Compression Methods for Multimedia Data

All of the compression techniques we've described so far try to reconstruct the data in full, without losing any information, so are called *lossless* compression methods. Another name for these techniques, because they can be described by information theory, is *entropy encoders*. When entropy compression routines were developed, most of the data in computing systems were programs or text data. This does not mean that these compression methods cannot be used for images, sounds, or videos. However, the compression obtained when using LZ77 or other variants is usually a factor two or less. There are many reasons for the lower compression rates, including:

- Entropy is usually higher for multimedia signals than for text data and differs across files, even if the multimedia data contains no noise (think of the alphabet needed to store an English text versus the alphabet needed to store an amplitude-modified clean sinus signal).
- Because sampled signals contain noise, it is almost impossible to find repeating patterns, such as is done in LZx algorithms.
- Multimedia data is usually multidimensional. To leverage redundancies between neighboring pixels or frames requires knowing the data's basic structure. For example, you must know the image's resolution to know what the neighboring pixels are.
- Many algorithms—for example, arithmetic coding—could work well with some multimedia content; however, their asymptotic runtime behavior makes using them for multimedia practically prohibitive.

The next chapters discuss methods for overcoming these challenges.

Index Terms

Exercises

1. Give one example of content for which run-length encoding (RLE) would work very well and one for which it would work pretty badly.
2. Specify three implementations of RLE that handle short run lengths differently. Discuss their advantages and disadvantages.
3. What is the information content of a coin toss?
4. Show that it is impossible to construct a compression algorithm that takes an arbitrary input string and always produces a shorter output string without losing any information.
5. Show that applying one compression method repeatedly does not yield significantly better results than applying it only once.
6. Create a tool that measures a file's entropy. Use the tool to calculate the entropy of three different text files containing English text, source code, a binary program, an uncompressed image file (for example, TIFF), and an uncompressed audio file.
7. Many file archival utilities provided in today's operating systems first concatenate a set of files and then apply compression techniques on the entire archive rather than on each file individually. Why is this method typically advantageous?
8. Discuss the efficiency of Morse code.
9. Construct the Huffman tree for the word "Mississippi."
10. Write a dynamic-tree Huffman encoder/decoder in a programming language of your choice. Don't forget to encode the tree.
11. What is the minimum number of bits needed to represent an arbitrary Huffman tree?
12. Would it make sense to combine Huffman encoding with RLE encoding? If yes, give an example where this would be useful.
13. Compress the word "Mississippi" using LZ77 and LZ78 as described in this chapter.
14. Discuss the usefulness of using an LZx algorithm to compress a video file (images only).
15. How would you compress an audio file using LZ77? Define a filter that would allow for better compression.
16. Compress the word "Mississippi" using arithmetic coding.
17. Give an example of a symbol distribution in which arithmetic coding could result in a shorter output string than Huffman coding
18. Implement a simple arithmetic coder using the Unix tool "bc."
19. Explain how arithmetic coding accounts for repeating words.

20. Which of the methods—RLE, Huffman, arithmetic coding, or LZW—would you use for the following files: a text file, an image file containing a screenshot, a photograph containing a portrait, a midi file, a sampled audio file containing rock music, a sampled audio file containing a generated sinus waveform, a movie (images only), and a cartoon animation (images only).

Literature

- David MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2006.
- Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann, 2nd edition, 2000.
- David Salomon, *Data Compression—The Complete Reference*, Springer, 2nd edition, 2000.

Web Links

- Compression Frequently Asked Questions:
  http://www.faqs.org/faqs/compression-faq/

Research Papers

- E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, July, October, 1948, pp. 379–423, 623–656.
- D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of IRE*, 1952, vol. 40, no. 10, pp. 1098-1101.
- J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, Vol. 23, No. 3, pp. 337-343.
- J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, 1978, vol. 24, no. 5, pp. 530-536.
- T.A. Welsh, "A Technique for High-Performance Data Compression," *Computer*, 1984, vol. 17, no. 6, pp. 8-19.
- I.H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, 1987, no. 30, vol. 6, pp. 520-540.

Chapter 14: Lossy Compression

Entropy-based compression as presented in the previous chapter is an important foundation for many data formats for multimedia. However, as already pointed out, it often does not achieve the compression rates required for the transmission or storage of multimedia data in many applications. Since compression beyond Entropy is not possible without losing information, that is exactly what we have to do: Lose information.

Fortunately, unlike texts or computer programs, where a single lost bit can render the rest of the data useless, a flipped pixel, or a missing sample in an audio file is hardly noticeable. Lossy compression leverages the fact that Multimedia data can be gracefully degraded in quality by increasingly losing more information. This is what will be explored in this chapter and the next chapters.

Mathematical foundation: Vector Quantization

Consider the following problem: We have an image that we want to store on a certain disk but no matter how hard we try to compress it, it won't fit. In fact, we know that it won't fit because information theory tells us that it cannot be compressed to the size of the space that we have without loosing any information. What choice do we have? Well, we could either not put the image on that disk or we could try to put as much of the image on the disk as we can fit, loosing some of the information.

There are multiple variants of how this could be performed: One could crop the image margins or just extract the parts of the images that are relevant to, let's say a customer. Of course, this would require an expert to be able to judge the relevance of each pixel.

A variant that can be performed automatically in a computer, without knowing anything about the data we are looking at is called vector quantization. Quantization is a general mathematical term and means discretizing a value range using different step sizes. Any given function f can be transformed into a quantized version q by using a transformation g. Often, a real-valued function f is converted to a integer-valued representation. For example:

$$Q(x) = g(\lfloor f(x) \rfloor)$$

Vector quantization involves vectors, i.e. instead of a single-valued function, tuples, triples, or n-dimensional vectors are quantized. In other words, vectors $[x_1,x_2,...,x_k]$ are to be mapped to val-

ues [y_1,y_2,...,yn] with n<<k. Think of these vectors as being anything, for example 8x8 pixel blocks in an image, red/green/blue triplets in a video, or 16 subsequent samples in an audio file.

So what is the best way to do it? The answer is: There is none but there are many to choose from! Which algorithm to use depends on the type and content of the data and the application. In the following we will present some common approaches.

Linear Quantization

The easiest way to map n vectors to k vectors with k < n is linear quantization. All of the n numbers are distributed evenly into k buckets. This is for example done by simple arithmetic rounding.

The problem with this method is that assuming an even distribution does not often yield not very good results because the distribution of the n numbers is not even. For example, consider an image where colors are to be reduced. Very often there may be a large amount of one color but not such a large amount of a second color.

K-Means

The K-means algorithm is well-known in statistics and machine learning as a clustering algorithm. This is an algorithm to partition n objects into k clusters, where k < n. The algorithm is based on the so-called expectation maximization principle. The idea of the algorithm is to minimize the intra-cluster variance, that is the squared distance between each member of a cluster to the mean value of each member of the cluster should be minimal. Formally, the function V

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

where there are k clusters S_i, i = 1, 2, ..., k, and μi is the centroid or mean point of all the points $x_j \in S_i$ is to be minimized. If the x_j are of higher dimension than 1, a different subtraction function has to be chosen, for example Euclidean distance.

The pseudo-code description of the code looks like this:

```
// Input: a set of data points X, an integer number k
// Output: a set of k data points representing the set μ
k_means(X, k)
    Choose k initial means μi from X at random
    WHILE (means μi are not updated significantly anymore)
        // maximization:
```

```
        FOREACH (sample xj: from X)
            assign membership of each element to a mean (closest mean)
        // expectation:
        FOREACH (mean μi)
            calculate a new μi by averaging xj that were assigned members
k_means := {μi}
```

Variants of the algorithm exist, where the means are not initialized at random but based on some assumptions or properties of the data. In practice, it is often hard to decide when the stopping criterion "not updated significantly anymore" is true. Therefore different criteria are used, often a fixed amount of iteration is preferred (loop n times). K-means is a very popular algorithm for all kinds of quantization tasks and is used in many fields as a "first guess" approach, i.e. try this one before you try anything more complicated. However, K-Means has three major limitations: The computational complexity increases dramatically with large amounts of data and, obviously, one has to know the amount of clusters k in advance. For example, the algorithm is well-suited for reducing 16777216 colors in an image to 65536 colors. But what if we don't know how many colors are really needed in the image to represent it well? For example, if the image contents is really just black and white but it is still represented by the aforementioned 65536 colors, we will use more colors than needed and waste memory, network bandwidths, or storage space. However, if we reduce a rainbow image to two colors, we might not be very happy with the appearance of the image in the end.

X-Means

An algorithms that tries to overcome the problem of having to know the number of clusters k in advance by guessing it is the X-Means algorithm. X-Means extends K-Means in several ways, including making the means computation more efficient by effectively caching computation results from previous iterations. The following section will discuss only the heuristics for automatically guessing k. The algorithms starts with a kmin which defines the lower bound of the range where k is to be searched and a kmax which defines the upper bound.

```
// Input: a set of data points X,
// kmin and kmax denoting lower and upper boundaries for k
// Output: a set of k data points representing the set μ
x_means(X)
     k := kmin
     Run k_means(X,k) until it converges
     Score the quality of the clustering
     IF (k>kmax) stop and report the best scoring clustering during the search
x_means := {μi}best
```

The main question with this algorithm is: How do we score the quality of the clustering?

Of course, we cannot do it optimally. If we could do it optimally we could greatly reduce the number of steps in this algorithm in the first place. Also, the optimal clustering depends on the underlying data we are processing and might be different for different types of image, video, and audio data.

However, the authors of the X-Mean algorithm, Pelleg and Moore, proposed the following heuristics that is based on general statistic assumptions. In order to score the quality of a clustering, each mean is split into two children: The children are moved a distance proportional to the size of the region in opposite directions. Next, in each parent region, a local K-Means with k=2 is run for each pair of children. Local means, the children are affected only by the points in the parent's region and not by any other parts of the data. Once this 2-means run has converged a test is performed on all pairs of children. Informally, the test asks: "is there statistical evidence that the two children are modeling a real structure here, or would the original parent model the distribution equally well?".

Many metrics have been defined in statistics to answer this question, none of them works perfectly, since again, the solution to this problem depends on the structure of data one is dealing with and the task one wants to ultimately accomplish. However, an often used metric that seems to work well in many cases is the so called Schwarz Criterion or Bayesian Information Criterion (BIC). In order to use BIC, one has to interpret the means as the mean of a spherical Gaussian [DEFINE IN APPENDIX] that defines a model to describe the data points belonging to the mean. This allows to assign a probability to each of the data points of belonging to the spherical Gaussian. We call, the Gaussian models M, so that M_j is the j-th model (derived from the j-th mean). BIC is then defined as:

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2} \cdot \log R$$

where l^j(D) is the log-likelihood [XXX DEFINE IN APPENDIX] of the data according to the j-th model, and p_j is the number of parameters in M_j (in our case the number p_j is the sum of the k-1 class probabilities, MK mean coordinates, and one variance estimate). R is the total number of data points which belong to the mean under consideration. BIC is a score, which basically favors the model with the minimum number of parameters. In other words: If our newly introduced two means represent the data equally well than one mean, we don't need to introduce two new means.

If BIC determines that the children means describe the data better than the parent, they are chosen instead of the parent (thus increasing k by one). Then the algorithm goes back to step one. Each time, a K-Means runs has converged, a global BIC score is calculated. When k is bigger than k_{max}, the globally computed BIC scores are compared and the best one, i.e. the highest, is chosen.

Perceptual Quantization

The mathematical methods described above are very useful for many tasks, especially if one wants to quantize data that one has no knowledge about other than general statistics. The main limitation of all of the methods described previously in this chapter and most of other methods is that it they tend to converge to local minima. Also, there is no way to tell whether they converged to a local minimum or not. Therefore, when applied to a concrete piece of data, the results can vary greatly, e.g. from one photo to another.

Fortunately, with acoustic and visual data we often have more background knowledge than just the theorems of mathematical statistics. As explained in Chapter XXX [introduction how ear works, etc...], we have a great deal of knowledge of how human perception works. Until the arrival of the digital age, the paradigm was that audio and video content should be reproduced as accurately as possible whenever copied. This means, that when comparing the original and the copy, the difference should be as small as possible. The idea behind perceptual quantization is that even though it is highly desirable to reproduce content as accurately as possible, the term "accurate" is not defined as a simple mathematical distance (such as an L-norm distance or root-mean-square error) but using some perceptual model. In other words: Two signal are accurately reproduced if the sound the same or they look the same even if they have many differences when compared bitwise. In order to achieve this, one needs a model of the perceptual sensors. The following sections describe the most important ideas of perceptual coding.

Sound Amplitude Quantization: A-law and μ-law

One of the simplest and yet most used perceptual quantization techniques for audio data are the A-law and μ-law algorithms. They area also referred to as companding algorithms, which is the older term inherited from analog signal processing. The idea is to exploit Weber-Fechner's law, which attempts to describe the relationship between the physical magnitudes of stimuli and the perceived intensity of the stimuli. As already explained in Chapter XXX, the Weber–Fechner law assumes that just noticeable differences are additive. As a consequence, sound intensity is perceived logarithmically. In other words, a sound must be twice as intense to be perceived a constant factor more intense in the human ear. The idea behind the A-law and μ-law algorithm is to

exactly exploit this fact: Rather than quantizing the intensity of the sounds, i.e. the sample values, linearly, they are quantized logarithmically. Thereby mapping slightly different intensities to the same value and loosing small sound intensity changes. Figure 1 shows the idea.

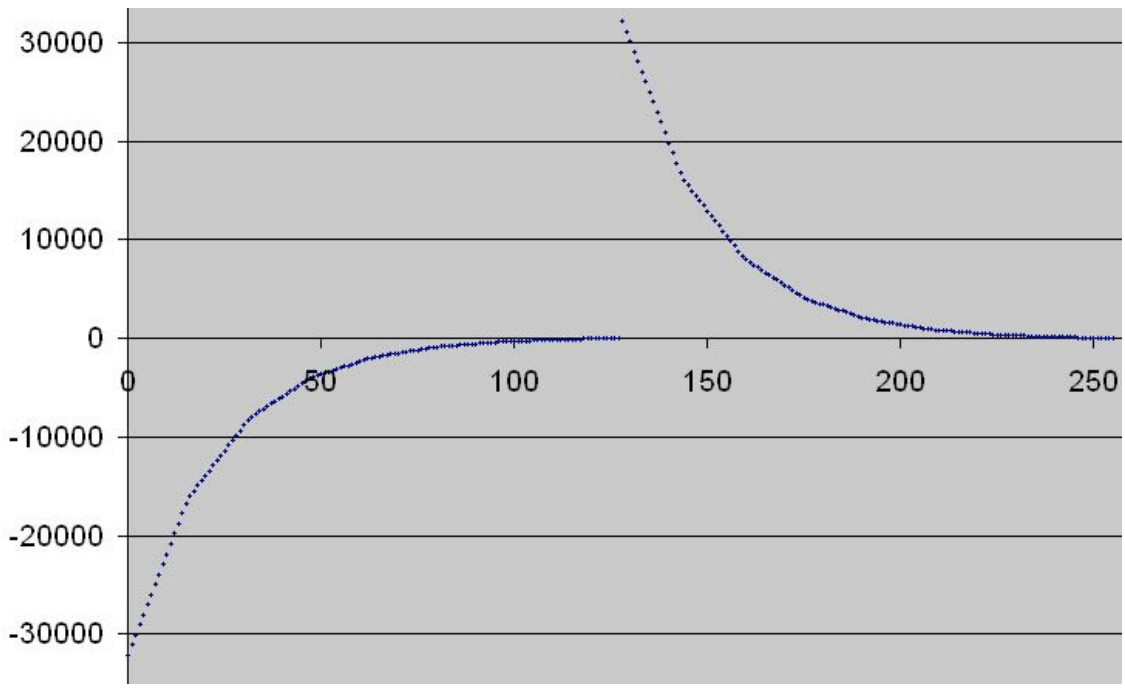


Figure 1. The μ-law quantization. The y-axis shows the input values and the x-axis shows the encoded values. One can see the logarithmic scale and the much denser concentration of values for low-amplitude input signals where the ear is more sensitive.

The companding formula for encoding a sample x normalized to the interval [-1,1] in μ-law is:

$$F(x) = \mathrm{sgn}(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \qquad -1 \leq x \leq 1$$

where μ is usually 255 for 8 bit and sgn(x) is the sign function. Uncompressing requires the inversion of the formula, which is:

$$F^{-1}(y) = \mathrm{sgn}(y)(1/\mu)[(1+\mu)^{|y|} - 1] \qquad -1 \leq y \leq 1$$

The companding formula for a-law is:

$$F(x) = \mathrm{sgn}(x) \begin{cases} \frac{A|x|}{1+\ln(A)}, & |x| < \frac{1}{A} \\ \frac{1+\ln(A|x|)}{1+\ln(A)}, & \frac{1}{A} \leq |x| \leq 1, \end{cases}$$

where A is the compression parameter. In Europe, A = 87.7; the value 87.6 is also used.

Decompression works as follows:

$$F^{-1}(y) = \mathrm{sgn}(y) \begin{cases} \frac{|y|(1+\ln(A))}{A}, & |y| < \frac{1}{1+\ln(A)} \\ \frac{\exp(|y|(1+\ln(A))-1)}{A}, & \frac{1}{1+\ln(A)} \leq |y| < 1. \end{cases}$$

Conceptually, both algorithms are the same. The μ-law algorithm provides a slightly better compression than A-law at the cost of worse proportional distortion for low-amplitude signals. Both of them are used world wide in digital telephony usually for compressing a 16-bit signal into a 12-bit signal. Since A-law is slightly better quality telephony companies agreed that A-law is preferred for an international connection if at least one country uses it. The μ-law algorithm has become a quasi standard for low-bandwidth voice recordings, as it is the default encoding of Sun's audio file format (file extension ".au"). This format is also the default format of Linux' */dev/audio* device and is supported by most common audio API's. It has been standardized as ITU-T Recommendation G.711. Both algorithms are practically implemented by one lookup table for encoding and one lookup table for decoding.

Visual Quantization

Like the ear, different quantities sensed by the eye scale logarithmically with intensity. One example is brightness. This had already been discovered by the ancient Greeks: Stellar Magnitude, which measures the light intensity of stars in the sky and was invited by Hipparchus in about 150 BC has a logarithmic scale. So by using a logarithmic scale for brightness, something like μ-law for audio could be effectively used for image data as well. A system that does this was patented for the first time by A. B. Clark of AT&T in 1928. Many different patents exist in this domain and image data can be quantized in many ways. Therefore there is no one algorithm standard for brightness quantization. We leave the creation of a grayscale image compression algorithm using a brightness compander as an exercise to the reader.

The old NTSC TV standard as well as the JPEG compression algorithm uses quantization in the different color channels. However, this is a different quantization since it involves spatial information.

Motion Quantization

Video itself is the best example for quantization at work. A video is a sequence of images that is replayed in a rapid succession. Just as any other sampling, a video is a quantized version of the reality it represents. The human eye is able to fuse the images into a moving scenes if the images are presented with about 1/30 of a second distance between them. In other words, we need a frame rate of about 30Hz to create the illusion of a moving scene. This, however, is just a rule of thumb. The actual frame rate depends on the physical state of an individual and on the content that is shown. The co-existance of a lip-synchronous audio track, generally allows for lower frame rates because the human brain is good at filling missing information across modalities. Therefore, when wanting to compress a video, it is often adequate to quantize the reality even further and play back a video at about 15 frames per second. A technique very often used for web demonstrations and other bandwidth-critical applications.

Differential Coding

Quantization methods are relatively simple to implement and offer a decent lossy compression scheme. However, they will usually not work anywhere beyond a compression ratio of 2:1 with acceptable perceptual reproduction quality. Differential coding is a scheme that may or may not build on quantization. It leverages global knowledge about the properties of the signal to encode. It is widely used for audio, image, and video compression alike and scales very well from lossless to entirely lossy. The general scheme is presented in Figure 2.

The main component of a differential encoder is the predictor. The predictor takes as input the n samples of the signal to predict the next m samples. The predicted samples are then compared to the original input. The difference is considered to be the encoding. If the prediction was perfect, only zeros would have to be transmitted to the decoder. Of course, nothing is ever perfect but good decoders will produce very small differences that do not need many bits. Different strategies are used to model predictors and the next section will present some of them. To decode the signal, previously decoded samples are used to feed an identical predictor. The prediction is then added to the encoding to reproduce the signal. To bootstrap the process, the first few samples can be predicted with 0 so that the actual signal is transmitted for initialization. This scheme by itself

is lossless. However, some predictors have a build in quantization and sometimes the output difference is thresholded. For errors not to become too large over time, some algorithms, alternate between lossy and lossless encoding, eg. every couple of frames, an unencoded sample has to be part of the stream. For better compression, difference encoding is often combined with entropy encoding.

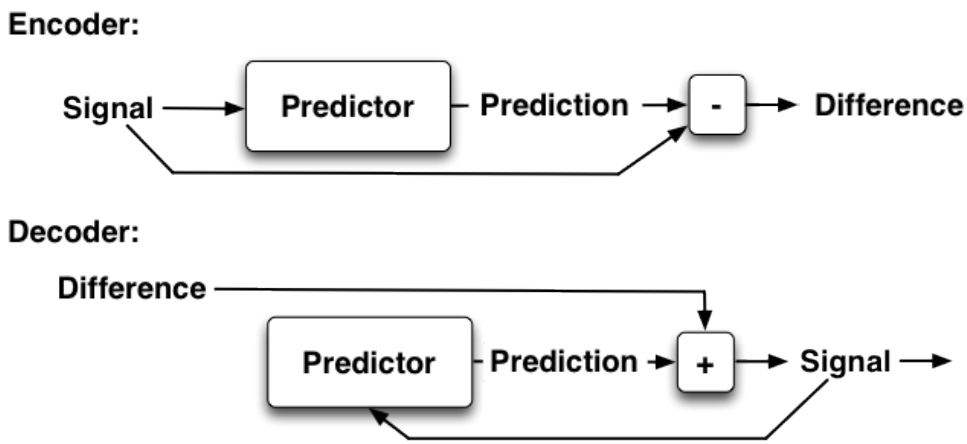


Figure 2. Schematic of a differential encoder: The signal is used as the input for a prediction model. The predictions are then subtracted from the original signal. The difference is transmitted. Decoding uses the previously decoded signals to feed the predictor which must, of course, be identical in both encoder and decoder.

Differential Coding in Audio: ADPCM

As a first example of a differential encoder, we look at ADPCM. ADPCM stands for adaptive differential pulse code modulation and is standardized in ITU-T G.726. It is a wide spread audio format and is used mainly for speech encoding. The idea behind the difference encoding of audio is that sound other than noise follows a rather predictable wave pattern i.e., the differences between consecutive samples will usually be much smaller than the samples themselves. Still, in some cases, sample values might drop from very high amplitude to very low amplitude and sometimes not. To maintain a constant-rate difference bitstream, one must furthermore predict whether these differences are large or not for the next couple of sample values. This, together with a quantization of the difference values is the idea of ADPCM.

The ADPCM algorithm is used to map a series of 12 bit μ-law (or a-law) PCM samples into a series of 4 bit ADPCM samples. Given the original sample values as input, ADPCM predicts both the next sample and the step size. A large step size means that the audio sample differences are large, a small step size means the differences are small. The IMA-ADPCM standard (Inter-

active Multimedia Association) defines ADPCM as shown in Figure 4. The 4-bit IMA ADPCM code consists of 4-bit: 1 bit for the sign and 3 bits for the difference.

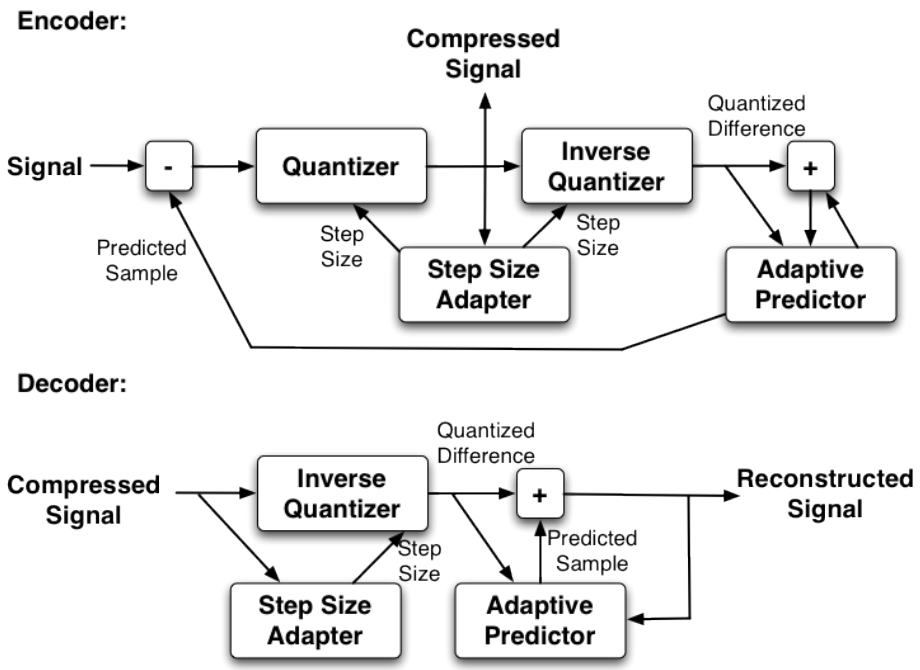


Figure 3. ADPCM encoder and decoder. A differential coder for audio signal that takes into account both the first and the second derivative of the signal. As with any predictive coding algorithms, the decoder block is embedded in the encoder.

Using a prediction model, an adaptive predictor guesses if future samples values might be large. If so, the algorithm increases the bits available for the difference encoding by adjusting the initial quantization. If it is determined that the step sizes might be smaller, the quantization is adjusted to a smaller bit length per sample. For every sample, two values have to be calculated: The difference to the previous (inverse quantized) signal sample and a value that determines the current step size. The process is bootstrapped by the first values all set to zero and the original sample being passed through.

Decoding works almost the same way: The compressed signal enters the decoder. The encoded value is the difference to the previously predicted value. This difference is added to the previous output value and constitutes the new output value. At the same time, the new step size is calculated which is needed to calculate the next prediction. Since decoder and encoder are very similar

and yet ADPCM, we will only provide the pseudo-code for a decoder here. For the concrete values of tables, header infomation, and magic numbers please refer to the standard itself.

```
// Input: a sequence of ADPCM-coded samples C
// Output: a sequence of raw audio samples S
adpcm_decode(C)
    Read header information from C
    First sample of S := first sample in C
    step_size_index := initial index provided in header
    stepsize := stepsizetable[step_size_index]
    oldsample := first sample in C
    FOREACH (sample c in C)
        delta := stepsize encoded in c
        s := oldsample + delta
        Add s to S
        step_size_index := indexTable[s]
        stepsize := stepsizetable[step_size_index]
        oldsample := s
decode_adpcm := S
```

ADPCM achieves a decent compression rate (about 4:1) and is very useable for speech signals. Music and other noise is not compressed very well using this methodology as the quality can suffer badly. It's not unbearable though so it could be used as a poor man's music compressor.

Differential coding in Images: PNG

The PNG image format uses a differential encoding step before an LZ-derivate entropy encoder is used. The algorithm predicts the color of each pixel based on the colors of previous neighboring pixels and subtracts the predicted color of the pixel from the actual color. An image line compressed in this way is often more compressible than the raw image line would be, especially if it is similar to the line above, since the differences from prediction will generally be clustered around 0, rather than spread over all possible image values. This is particularly important in relating separate rows, since the later applied entropy compression (DEFLATE see Chapter XXX), has no understanding that an image is a 2D entity, and of course interprets the image data as a stream of bits. The predictor uses the pattern depicted in Figure 3 to sift through the image.

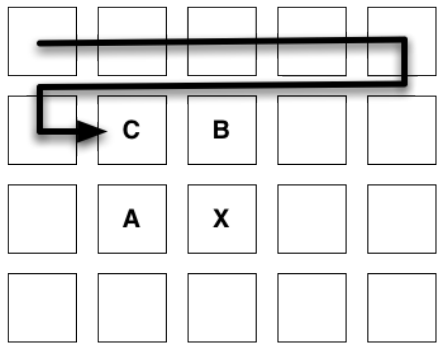


Figure 3. The PNG image is scanned line by line. The color value of pixel x is the one to be predicted and one of five different predictor states can be used that depend on the color values of a, b, and c.

The predictor has five states which predict the value of each byte (of the original image data) based on the corresponding pixel to the left (a), above (b), above and to the left (c) or some combination thereof, and encodes the difference between the predicted value and the actual value. The states are shown in Table 1:

State	Name	Predicted value
0	None	Zero (so that the raw value passes through unaltered)
1	Sub	Value of a (to the left)
2	Up	Value of b (above)
3	Average	Mean of bytes a and b, rounded down
4	Paeth	a, b, or c, whichever is closest to p = a + b − c

Table 1. Predictor states used for differential encoding in the PNG image format.

The Paeth filter computes a simple linear function of the three neighboring pixels (a, b, c), then chooses as predictor the neighboring pixel closest to the computed value as defined by the following pseudo-code:

```
// Input: color values a,b, and c as illustrated in Figure 3
// a = left, b = above, c = upper left
// Output: a paeth-prediction for a,b, and c
paeth_predict(a,b,c)
    p := a+b-c
    pa := abs(p-a)
```

```
    pb := abs(p-b)
    pc := abs(p-c)
    IF (pa<=pb AND pa<=pc) p := a
    ELSE IF (pb <= pc) p := b
    ELSE p := c
paeth_predict := p
```

Compression of a pixel value x dependent on its neighbors a,b, and c works by calculating

$$compressed(x) = x - paeth\_predict(a,b,c)$$

and decompression works by reversing the formula to

$$uncompressed(x) = compressed(x) + paeth\_predict(a,b,c).$$

The particular states are chosen adaptively on a line-by-line basis based on a heuristic developed by Lee Daniel Crocker, who tested the methods on many images during the creation of the format. If interlacing is used, each stage of the interlacing is filtered separately. It makes the compression generally less effective though.

Differential Coding in Video: Motion Compensation in MPEG-1

Differential encoding is also used for videos. Every video that is shipped on Video CD, DVD, Blue Ray Disk, or through digital cable, satellite, or antenna television is compressed using an algorithm that contains a differential encoder. The idea is that at 25 frames per second or more, the differences between two or even more consecutive video frames are minor. Any object in the camera view that does not change its appearance or position during 0.04 seconds will not have changed in between two frames. Also, physical objects tend to not randomly change position. In other words, an object that is visible on the left and is known to have changed its position rightwards over the last couple of frames will probably either stop in the next frame or continue to do so. This is the idea behind motion compensation: Rather than just encoding video frames as stand-alone images, the difference between them is modeled by a differential encoder. This technique is used in different variations in all versions of MPEG video and in many other video codecs. In the following, we will describe one version of the MPEG-1 motion compensation.

The MPEG-1 algorithm works on a block-by-block basis. In the next chapter we will explain what the advantages of this approach are. For now, it is important to know that an 8x8 pixel block is called macro block. To decrease the amount of spatial redundancy in a video, only

macro blocks that change inside a certain amount of consecutive frames are updated. This is known as conditional replenishment. However, conditional replenishment is not very effective by itself. Movement of large objects, and/or the camera may result in large portions of macro blocks needing to be updated, even though only the position of the previously encoded objects has changed but not their appearance. Through motion prediction the encoder can compensate for this movement and remove a large amount of redundant information. The encoder compares the current frame with adjacent parts of the video from the previous frame up to an encoder-specific predefined radius limit from the area of the current macro block. If a match is found, only the direction and distance (i.e. the vector of the motion) from the previous video area to the current macro block need to be encoded.

Of course, a predicted macro block rarely matches the current frame perfectly. The differences between the predicted matching area and the real macro block is therefore the lossy part of the process. The larger the error, the more data must be additionally encoded in the frame.

The distance between two areas in a frame is measured in number of pixels but is often referred to as pels. MPEG-1 video uses a motion vector precision of one half of a pixel or half-pel. The finer the precision, the more accurate the match is likely to be, and the more efficient the compression. Higher precision, however, also requires higher runtime of the encoder and also a larger encoding bitrate since potentially more motion vectors have to be stored. In the end, since neighboring macro blocks are very likely to have similar motion vectors, only the difference between the motion vectors has to be encoded for each macro block. The pseudo-code for a very simple, exhaustive motion estimator is shown below:

```
// Input: REF = reference frame, CUR = current frame
// Output: K a set of indizes to the 8x8 blocks in REF most
// closely the matching the block in CUR
motion_estimate(REF, CUR)
    FOREACH (block MB in CUR)
       FOREACH (i := 0,1,...,64) //8x8 block
           pcur[i] := pixel in MB
           pref[i] := pixel in REFMB
           FOREACH (block k in REF)
               distortion[k] := SUM(distortion(pcur[i],pref[k,i])
       Add minimum value for distortion[k] to K
motion_estimate := K
```

Ideally, the function distortion reflect human perception. Unfortunately, this is still a matter of research, therefore very often Euclidean distance or other similarly simple metrics are used.

In order to bootstrap the process, the first frame is not dependent on any other frame. It is just a JPEG image (so-called I-Frame). The frames after the I-Frames are called P-frames (predictive frames). MPEG also defines B-frames. These frames takes into account the previous and the next frame for motion prediction and can therefore achieve higher compression ratio. However, it is necessary for the player to first decode the next frame sequentially after the B-frame, before the B-frame can be decoded. Therefore B-frames are computationally more complex both in encoding and decoding.

Since motion compensation is lossy and each frame depends on the previous one the error propagates easily and grows with every frame. Therefore, and also for the ability to play a video from a random time position, I-frames are inserted every couple of frames. The I-frame and all frames that ultimately depend on it, are called group of pictures (GOP). A typical GOP size is about 20 frames.

Except ADPCM, differential encoding is rarely used stand alone. Both MPEG and PNG rely on additional compression steps. The next chapter is going to explore the ideas behind them and also other, more advanced, techniques.

Index Terms

Exercises

1. Implement linear quantization and K-means and use the implemented quantization algorithms to reduce the colors of several color images to 2, 4, 8, 16, 64, and 256 colors. Which algorithm performs "best"?
2. Try linear quantization on an audio file with different granularity. What artifacts can you hear?
3. Find or implement a μ-law compression and decompression table. Modify the algorithm such that it compresses down to 4 bits.
4. As described in the text, create a logarithmic compander for light intensity and try it on gray scale images.
5. Try to apply μ-law compression as a color quantization algorithm for an image.
6. Apply an entropy encoder of your choice to a set of companded audio files and compare the compression of applying the same entropy encoding to the uncompanded files. How do you explain the results?
7. Create a simple predictive coder and decoder for a sampled audio file only based only on the difference between two sample values. This algorithm is often referred to as differential pulse code modulation (DPCM).

8. Change your DPCM encoder so that instead of trying to minimize the amount of bits for each sample, it maximizes the amounts of bits (probably even taking more bits than the original). Make sure, it is still a differential compression scheme (although a maximally bad one).
9. Design a differential encoder/decoder for ASCII text files containing natural language. Try to model your predictor so it takes into account the properties of the natural language the original text is encoded in (e.g. the frequency of the individual characters following other characters).
10. Discuss the properties of sound files when ADPCM compression will work well and when it won't. Both compression efficiency and perceptual accuracy should be described.
11. Would the application of a smoothing algorithm help improve the quality of the output of the ADPCM encoder?
12. Implement the PNG difference encoder as described in the text and apply it to some test images. Describe how you choose the heuristics to find the state of the predictor.
13. Add another predictor state to your PNG algorithm -- when would you choose it and why do you think it is good?
14. Extract several consecutive frames from a video and save them as JPEG.
15. a) Calculate the difference image and discuss why certain pixels are shown in the difference image.
16. b) Using several frames from the video, implement a routine to calculate the motion vectors for 8x8 blocks.

Literature

- Ken C. Pohlmann (1985). Principles of Digital Audio (2nd ed.). Carmel, Indiana: Sams/Prentice-Hall Computer Publishing. ISBN 0-672-22634-0.
- μ-law: ITU.T Recommendation G.711
- ADPCM: ITU.T Recommendation G.726
- PNG: RFC 2083
- MPEG-1 Video: ISO/IEC-11172-2
- Paeth, A.W., "Image File Compression Made Easy", in Graphics Gems II, James Arvo, editor. Academic Press, San Diego, 1991. ISBN 0-12-064480-0.

Web Links

- Quantization threads in comp.dsp:
  http://www.dsprelated.com/comp.dsp/keyword/Quantization.php
- Quantization links in datacompression.info:
  http://datacompression.info/Quantization.shtml
- PNG home:
  http://www.libpng.org/pub/png/

Research Papers

- Ralph Miller and Bob Badgley: U.S. Patent 3,912,868 filed in 1943: N-ary Pulse Code Modulation.
- Franklin S. Cooper; Ignatius Mattingly (1969). "Computer-controlled PCM system for investigation of dichotic speech perception". Journal of the Acoustical Society of America 46: 115. doi:10.1121/1.1972688.
- P. Cummiskey, N. S. Jayant, and J. L. Flanagan, "Adaptive quantilation in differential PCM coding of speech," Bell Syst. Tech. J., vol. 52, pp. 1105—1118, Sept. 1973.
- J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297.
- Dan Pelleg, Andrew Moore: "X-means: Extending K-means with Efficient Estimation of the Number of Clusters", Proceedings of the Seventeenth International Conference on Machine Learning, pp. 727-734, San Francisco, 2000.
- Didier Le Gall: "MPEG: a video compression standard for multimedia applications", Communications of the ACM, vol. 43, no 4, pages 46-56, April 1991.

Chapter 15: Advanced Perceptual Compression

The lossy compression techniques presented so far have tried to exploit the fundamental mathematical properties of information (lossless coding), to model and approximate the properties of the signal directly (differential coding), and to model the creation of the signal (source coding, such as in the speech compression). We also presented simple perceptual methods, such as the μ-law encoder.

The methods presented in this chapter use transformations that have been modeled after how human sensory perception works using a much grater sophistication-level. These perceptual coders are so effective, that they are used in virtually every device today that handles images or sound: From photo cameras to mobile phones to DVD players to mobile digital music players.

Before we start introducing them, we recapitulate a fundamental signal transformation which is an important prerequisite for all the algorithms presented in this chapter as well as for many of the analysis algorithms presented later on. When explaining perceptual compression, two transformation are outmost important: The Discrete Fourier Transform (explained in Chapter XXX [basic audio processing]) and the Discrete Cosine Transform, which is described in the following sections. Other transforms, such as the Discrete Wavelet Transforms which are a generalization on the transforms mentioned, are also used in multimedia signal processing and the references cited below are well worth looking up.

Discrete Cosine Transform (DCT)

The discrete cosine transform (DCT) expresses a sequence of finitely many data points as a sum of cosine functions, rather than sine functions in the DFT, oscillating at different frequencies. The most important difference between DCT and DFT for practical applications is that the DCT is using only real numbers. The DCT is equivalent (up to an overall scale factor of 2) to a DFT of $4N$ real inputs of even symmetry where the even-indexed elements are zero. This property allows to apply the transformation without ever increasing the number of elements in the sequence. There are some variants of the DCT which differ slightly in property, but the most common one is given in the following equation. The N real numbers $x_0, ..., x_{N-1}$ are transformed into the N real numbers $X_0, ..., X_{N-1}$ according to:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1.$$

The inverse DCT (IDCT) is then given by multiplying the DCT with

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N}n\left(k+\frac{1}{2}\right)\right] \qquad k = 0, \ldots, N-1.$$

Figure 1 shows a visualization of the DCT coefficients given an input signal. Again the DC coefficient is determining the overall gain of the signal, while the AC coefficients represent the energies of the signal in different the frequency bands.

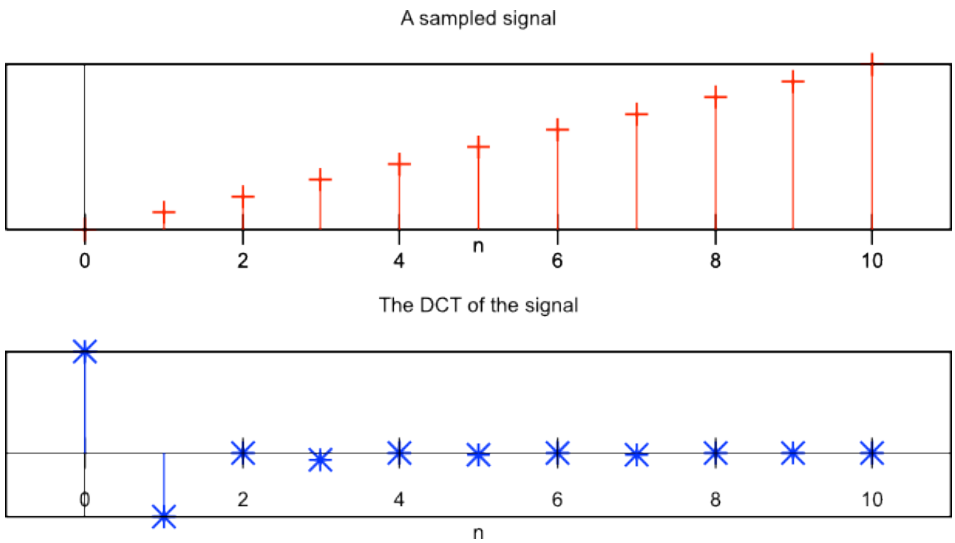


Figure 1. Example DCT transform of a small sequence of samples.

To compute the DCT, DFT, inverse DCT, and inverse DFT on multidimensional data, i.e. matrixes or N-dimensional data, all one needs to do is compose the DCTs and DFTs computed along each dimension. A two-dimensional DCT, for example, is therefore given as:

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right].$$

The following two pseudo-code snippets compute a two-dimensional DCT.

```
// Input: A number N
// Output: An NxN DCT matrix H.
dct_matrix(N)
    FOR i:=0 TO N
          FOR j:=0 TO N
                H[i][j]:=sqrt(2/N)*cos(pi*i*(2*j+1)/(2*N))
    FOR j:=0 TO N
          H[0][j]=H[0][j]/sqrt(2)
dct_matrix := H
```

```
// Input: N, an NxN matrix of samples M, a DCT matrix H
// Output: The DCT coefficients of M, called C.
DCT(N,M,H)
    // Transform columns
    FOR k:=0 TO N
          FOR l:=0 TO N
                sum:=0
                FOR m=0 TO N
                      sum:=sum+H[k][m]*M[m][j]
                C[k][l]:=sum
    // Transform rows
    FOR k:=0 TO N
          FOR l:=0 TO N
                sum:=0
                FOR m=0 TO N
                      sum:=sum+H[l][m]*M[k][m]
                C[k][l]:=sum
dct_matrix := c
```

The code shown above is not very efficient as it does no reuse of intermediate results. However, it shows the strategy of pre-computing the matrix, which has advantages when experimenting with different variations of the function.

Figure 2 shows combination of horizontal and vertical frequencies for an $N=8$ two-dimensional DCT. Each step from left to right and top to bottom is an increase in frequency by 0.5 cycle, e.g. moving right one from the top-left square yields a half-cycle increase in the horizontal frequency. The picture provides an initial intuition for why the DCT is useable for image compression: A given 8x8 square in Figure 2 is imaginable to be a part in a black and white image. In other words, any given 8x8 pixel square in an image can be (lossily) reduced to one of the 8x8 blocks in Figure 2. This is exploited in a very common image compression algorithm, commonly referred to as JPEG, which is explained in the next section.

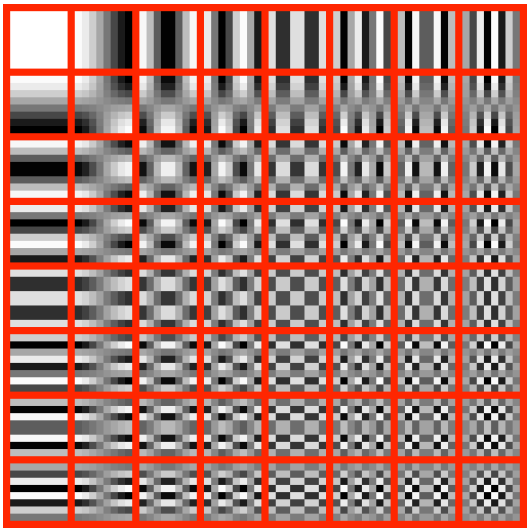


Figure 2. DCT frequencies for 8x8 matrixes in X- and Y- increasing order.

JPEG

JPEG is the single most common image format used by digital cameras and other photographic image capture devices. The name "JPEG" stands for Joint Photographic Experts Group, the name of the committee that created the standard. The group was organized in 1986, issuing a standard in 1992, which was approved in 1994 as ISO 10918-1. The file format is named “JFIF” JPEG File Interchange Format but the acronym jpeg or jpg is used more commonly both as a file extension as well as a name of the file format. Apart from JFIF there is also another standard, called “EXIF” (Exchangeable Image Format) which on top of the image content also standardizes the storage of specific metadata, such as geolocation and camera manufacturer. The following section presents an overview of the image compressionn part of the algorithm. The encoding uses several steps, of which the most important is a quantization in frequency space, generated by the Discrete Cosine Transform.

The following pseudo-code outlines the algorithm:

```
// Input: A color image I in 24-bit RGB format
// Output: A JFIF-compressed image J.
JPEG(I)
    transform color space from RGB to Y,Cr,Cb
    split I into 8x8 blocks for each Y, Cr, Cb
    scale down the resolution of the Cr and Cb blocks to 4x4 blocks
    Apply DY:=DCT(Y), DU:=DCT(Cr), DV:=DCT(Cb).
    Quantize DY, DU, and DV according to a user setting
    Linearize DY, DU, and DV into a stream S
    Apply static Huffman(S)
```

```
    J := Header information + S
JPEG := J
```

In the following, we will now describe each step and why it is applied.

Tiling

The first step is to split the image into 8x8 pixel blocks. The splitting was originally performed to make the computation of the DCT more tractable. On today's computers this is not an issue anymore and a larger block size, such as 16x16 is sometimes applied. Furthermore, the splitting allows for very efficient and easy parallelization on manycore architectures.

Chroma Quantization

Then, any given pixel is converted to the Y,Cb,Cr color space (color spaces are described in Chapter XXX). This allows to apply a technique called chroma quantization. The idea is that because the human visual system is less sensitive to the position of color than brightness, bandwidth can be saved by storing more detail about the brightness of a pixel than of the color. At normal viewing distances, there is no perceptible loss incurred by sampling the color detail at a lower rate. The signal is divided into a luma (brightness) component Y and two chroma (color difference) components Cr, Cb (see Chapter XXX [color spaces]). The subsampling scheme is commonly expressed as a ratio R:A:B (e.g. 4:2:2) that describe the number of luminance and chrominance samples in a conceptual region that is R pixels wide, and 2 pixels high. R is the width of the region (e.g. 4 pixels), A is the number of chrominance samples (Cr,Cb) in the first row of R pixels, and B is the number of chrominance samples (Cr,Cb) in the second row of R pixels. The old analog TV standard NTSC, for example, used a fixed 4:1:1 chroma quantization scheme.

Frequency Space Quantization

The human eye is able to perceive small differences in brightness over a relatively large area. However, high frequency brightness variation are not easily distinguishable by human visual perception. In other words, details in a texture may be blurred without reduction in perceived image quality. JPEG utilizes this fact by reducing the amount of information in the high frequency components. Therefore, each block is converted into frequency space by applying a DCT. Then each coefficient in the block is divided by a constant for that component, rounding to the nearest integer. The division constants are specified in a so-called quantization matrix, which is varied depending on a user-definable quality factor. A typical quantization matrix looks like this:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Note, that, apart from the DC coefficient in the upper left corner, the lower frequencies have smaller quantization and the higher frequencies have higher quantization. When the user wants higher quality, the quantization values can be lowered, when higher compression is a goal the quantization coefficients can be increased in value. Figure 3 shows an image with different strengths of quantization applied.

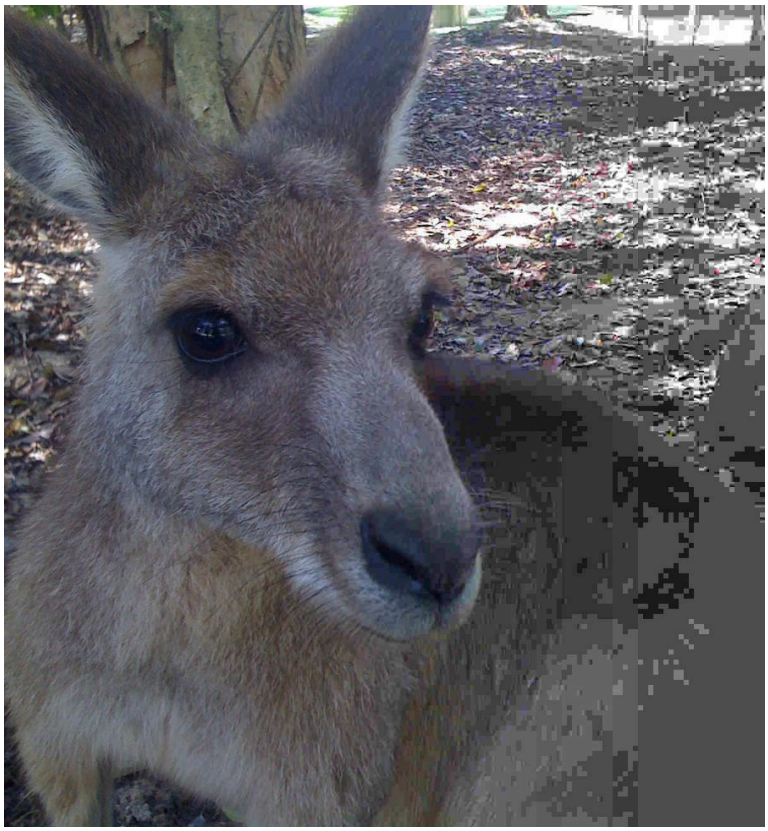


Figure 3. Sample JPEG image with increasing amount of quantization (increasing from left to right)

As a result of applying quantization to the DCT blocks many of the higher frequency components are typically rounded to zero, and many of the rest become small positive or negative numbers, which take fewer bits to store.

Linearization and Lossless Encoding

After quantization, many of the higher frequencies components of a block are zero or close to zero. Therefore, the final step is to use run-length encoding, followed by a Huffman compressor (see Chapter XXX). In order to apply run-length encoding, the blocks are linearized using a scheme that allows the same frequencies to be in sequences, which makes many 0s and the potentially same values in X- and Y- direction appear in sequence. Figure 4 shows the linearization scheme.

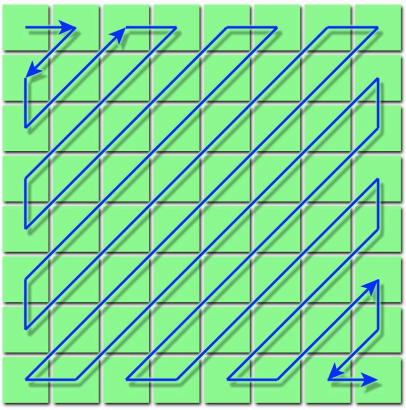


Figure 4. Linearization scheme for an 8x8 JPEG block.

Depending on the image and chosen quantization compression rates of 1:2-20 and higher compared to an uncompressed image are possible. Decompression works by reversing the process, eg. multiplying with the quantization matrix, using the inverse DCT, and so on. Figure 5 shows a comparison of compression and decompression.s

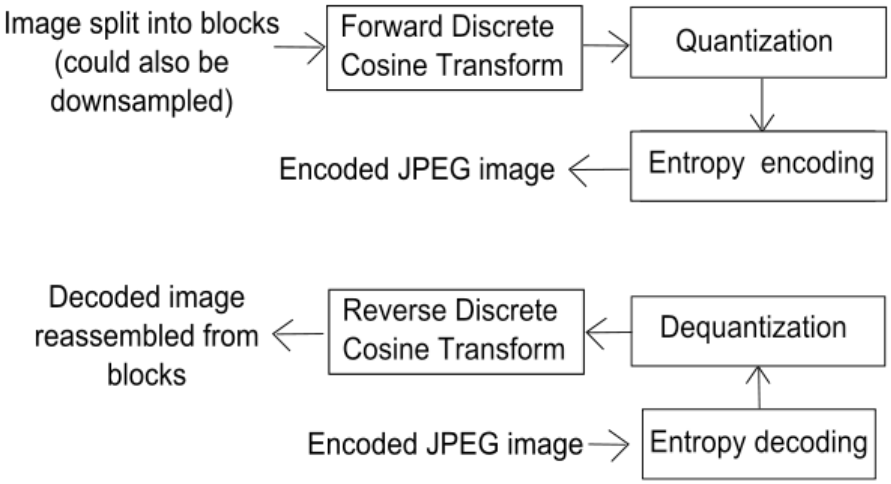


Figure 5. Overview of the JPEG compression/decompression algorithm.

Newer versions of the JPEG standard, e.g. JPEG2000, use different transforms, such as Wavelet transforms (see references) instead of the DCT to reduce artifacts after quantization in higher compression modes. Also, some implements allow the use of arithmetic coding instead of Huffman coding. Many vsideo codes have adopted the JPEG encoding methods but sometimes vary the block size and the linearization scheme.

Psychoacoustics

Similar to the exploitation of brightness versus chroma perception and high-frequency blurring in JPEG, audio signals can be compressed based on human auditory perception. We already described audio codecs that utilize perceptual properties of the human ear, such as the μ-law compression in Chapter XXX that utilizes the non-linear perception of amplitude. Like JPEG, audio codecs can leverage perceptual properties to determine which parts of the signal can be quantized more aggressively or even removed without changing a human's perception of the result. Unfortunately, the human ear is more sensitive to signal alterations than the human eye. So applying the JPEG scheme of transforming the signal into the frequency domain and then linearly scaling the coefficients would work for compression but would result in significant audible artifacts. To achieve acceptable quality and compression at the same time, one has to take into account a wider range of perceptual properties of the human auditory system. As a result, the development of modern audio compression methods is mostly one of measuring and estimating how various sounds are perceived by people, a field known as psychoacoustics. The resulting lossy compression schemes, however, routinely lead to audio files that are about 10% the size of high quality masters with very little discernible loss in quality. Formats based on psychoacoustics include

Ogg Vorbis, Dolby AAC, Microsoft WMA, MPEG-1 Layer II (used for digital audio broadcasting in several countries), and Sony ATRAC. In the following, we will provide an overview of the functionality of one version of the most prominent representative of perceptual audio codecs: MPEG-2 Audio Layer 3 or short MP3.

We start with the presentation of the most important psychoacoustic phenomena utilized by audio codecs.

Frequency/Loudness Resolution

As explained in Chapter XXX, the human ear perceives sounds in the range of about 16 Hz to 20 kHz. The smallest pitch differences a human can perceive depends on the absolute pitch values, but is reported based on clinical experiments to be about 3.6 Hz within the frequency range of 1000-2000 Hz. However, smaller pitch differences may be perceived through the interference of two pitches, such as an effect that is known as "beating": The phase variance caused by two interfering frequencies might be heard as a low-frequency difference pitch.

The intensity range of audible sounds is enormous but also depends on the frequency. Roughly speaking, the lower the frequency the louder a sound is perceived. This is expressed in different scales that have been proposed in the literature derived from psycho-acoustic experiments, such as the Mel scale and the Bark scale. The Bark scale was proposed by Eberhard Zwicker in 1961 and is named after Heinrich Barkhausen who proposed the first subjective measurements of loudness. It seems to be more popular with music processing while the Mel scale is used more often in speech processing. Also, the Bark scale is used for equal loudness assumptions in audio codecs. It is approximated by the following equation:

$$Bark = 13\arctan(0.00076 \cdot f) + 3.5 \cdot \arctan\left(\left(\frac{f}{7500}\right)^2\right)$$

with f being the frequency of the tone in Hz. Sounds of equal Barks are estimated to have equal loudness. Figure 5 shows a plot of the Bark scale.

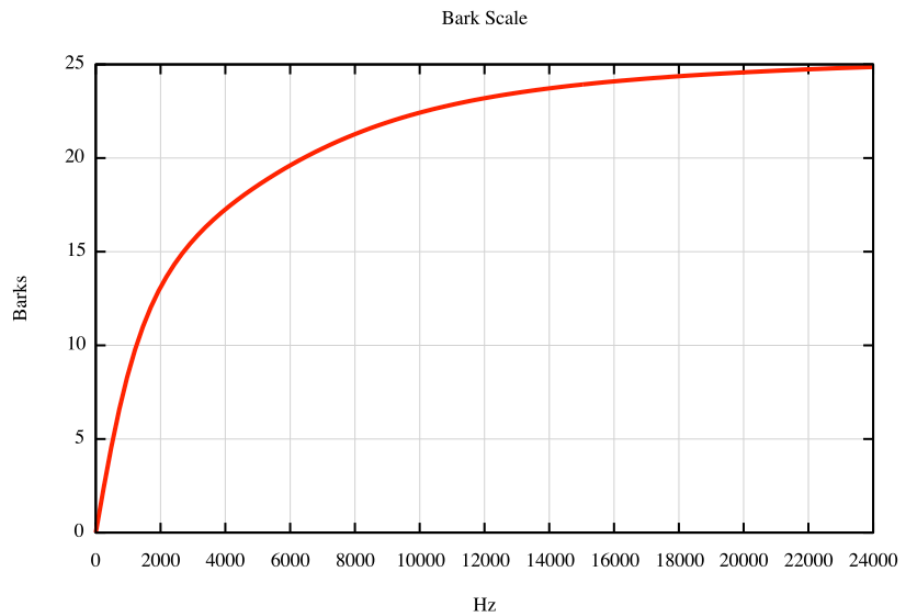


Figure 5. An example of an equal loudness scale curve: The Bark scale.

Very important for compression are the lower limits of audibility. This so-called absolute threshold of hearing (ATH) curve is derived by exposing humans to testing tones of various frequencies. Typically, the ear shows its lowest ATH between 1 kHz and 5 kHz (the range of speech), though this threshold also changes with age, with older ears showing decreased sensitivity above 2 kHz. ATH is usually approximated by the following formula (from Terhardt 1979):

$$T_q(f) = 3.64\left(\frac{f}{1000}\right)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}\left(\frac{f}{1000}\right)^4$$

Figure 6 shows the resulting curve.

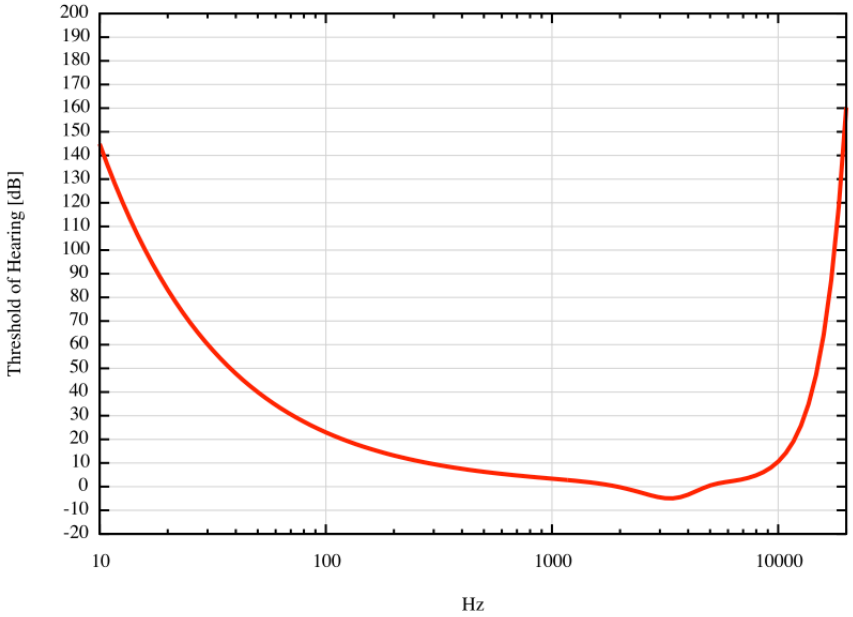


Figure 6. An estimate of the average lower hearing threshold (ATH).

Masking

While the two properties studied above were done on measurements of isolated sounds, masking describes a property of the ear that concerns the interaction of different sounds: An otherwise clearly audible sound can be masked by another sound and thus become imperceptible.

The phenomenon can be experienced very easily. For example in the situation where two people have a conversation at a bus stop. Suddenly a truck drives by and due to its noise the conversation is interrupted: The voices have become imperceptible in the presence of the truck noise. A weaker sound is called to be masked if it is made inaudible in the presence of a stronger sound. The phenomenon occurs because the loud sound distorts the ATH, making the quieter sounds (partially) fall below the threshold.

Masking can occur simultaneously (as in the bus-stop example) and sequentially. A quieter sound emitted very soon after the decay of a stronger sound is masked by the louder sound. Also, a quieter sound just before a louder sound may be masked by the stronger sound. These two effects are called forward and backward temporal masking, respectively. Figure 7 illustrates the phenomenon of masking. An important property that determines how strong a particular sound can mask another one is its tonality. For example, a sinusoidal masker requires a higher intensity to mask a noise-like “maskee” than a loud noise-like masker does to mask a sinusoid. Quantization

can be seen as introducing noise and thereby masking good sounds with bad sounds. Psycho-acoustic models used for compression take this into account by quantizing pure tones less than already noisy ones. This is mostly accomplished by binning the frequency bands appropriately and quantizing frequency bins with higher energy less than those with lower energy.

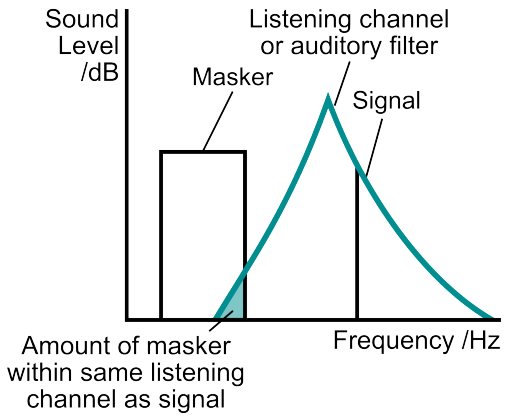


Figure 7. Illustration of Masking: A signal is masks large parts of the frequency space, rendering sounds in it (partially) imperceptible (adapted from Moore 1998).

In addition to frequency and temporal masking of sounds, sounds are also masked spatially. This means, a sound perceived in one ear might prevent perception of sounds arriving at the other ear. This phenomenon is exploited to reduce perceptual redundancies across channels when stereo coding.

MP3

There are many implementations of MP3 encoders and they differ in quality, compression rate, and many technical details. The ISO standards (see references) only describes the methods needed to decode a stream, which are much simpler, as will be explained later. Encoders usually reduce perceptual redundancy by first identifying sounds which are perceptually irrelevant, such as high frequencies and sound levels below ATH. Unfortunately, the reduction of imperceptible portions of sounds in an audio file usually only accounts for a small percentage of the bits needed to represent the signal. Therefore, like in JPEG, the more effective method is to quantize the signal.

As already mentioned earlier, a simple linear quantization with linear frequency bins, such as frequently applied in image and video compression, would not yield high enough quality as the ear is more sensitive to artifacts than the eye. The method used for audio is called noise shaping. Reducing the number of bits used to code an audio signal increases the amount of noise in the signal. The idea of noise shaping is to hide the noise generated by the quantization in areas of the

audio stream that is least perceived, based on masking tables, ATH, and the Bark scale. Noise shaping can also be understood as distributing a predefined number of bits to the different frequency bands according to their perceptual priority given a psycho-acoustic model. For example, higher frequencies of the signal might be coded with less bits as it is harder to perceive small differences in these regions. If reducing perceptual redundancy does not achieve sufficient compression for a particular application, it may require further lossy compression. Depending on the audio source, this still may produce acceptable perceptible quality. Let's look at the concrete implementation of an MP3 encoder, the open source compressor LAME.

Algorithm

Similar to the tiling in JPEG, the first operation on the audio signal is splitting it into packets, so called granules (sometimes also called frames or chunks). LAME uses a granule size of 576 samples. These are then transformation into frequency space. This is usually done using a modified version of the DCT, called MDCT (see references). Once in frequency space, the coefficients are split into 32 different equal-sized frequency bands (each of them roughly 700Hz). For each granule, the algorithm then reserves a certain amount of bits usually pre-defined by the user. Constant bitrate (CBR) encoders allow only a certain maximum amount of bits per granule, average bitrate encoders (ABR) let the encoder try to achieve an average bitrate over the entire stream and variable bitrate encoders, enable maximum audio quality by allowing a variable bitrate on the stream.

The following pseudo-code describes the operation of LAME's so-called "outer loop", which is the algorithm that finds the combination of quantization coefficients (here called scalefactors) to produce the least amount of audible distortion (cite LAME website).

```
  // Input: A number of MDCT coefficients C binned in frequency bands.
  // Output: A set of scalefactors S.
  MP3_outerloop(C)
      S[]=0
      DO
            compute better quantization using S (call inner_loop)
            compute distortion within each scalefactor band
            compare distortion to allowed distortion (from psy-model)
            over := number of bands where distortion > allowed_distortion
            tot_noise :=
            average over all bands of distortion(db) - allowed_distortion(db)
            over_noise :=
                see tot_noise but only bands with distortion > allowed_distor-
tion
            IF this quantization takes the least bits so far, save it in S.
            IF over=0 THEN return S.
            reduce quantization (use more bits) for bands with distortion
      WHILE (over>0) OR NOT (all scalefactors set to their max)
```

```
MP3_outerloop := S
```

In order to find the "better" quantization scalefactors in each iteration, both noise shaping and the bitrate specified by the user are taken into account in the "inner_loop". The algorithm performs an exhaustive search over all possible values for the quantization factors, measuring both the resulting bitrate and the resulting perceptible noise as given by the psychoacoustic model and choses the optimal configuration. This search is what takes most of the runtime of an MP3 encoder.

When a stereo signal is to be encoded, Lame knows two stereo modes: stereo and joint stereo. The stereo modes treats the left and right channel completely independently. Joint stereo means individual audio frames may be encoded in either normal stereo or mid/side stereo.

Mid/side stereo encodes the stereo signal into two channels, the middle channel which contains the sum of both the left and right channel and the side channel which contains the channel differences between the middle channel and the left and right channel, according to this scheme:

$$Middle = \frac{L+R}{2} \qquad Side = \frac{L-R}{2}$$

with *L*=left channel and *R*=right channel. For decoding, *L=Middle+Side*, *R=Middle-Side*. More bits are allocated to the middle channel than to the side channels. Audio signals where the stereo channels are not well separated have will have very little information in the side channel. Therefore this technique will improve bandwidth. However, there will be little gain for well-separated channels and any encoding errors in this mode will show up as noise in *both* the left and right channels after decoding. Therefore LAME choses to switch between mid/side stereo and regular stereo on a frame-by-frame basis based on the difference in masking thresholds between the right and the left channel, i.e. when LAME decided that the perceptual difference between the left and right channel is less than 5dB, it uses joint stereo.

After bitencoding the quantized coefficients, a Huffman encoder is used to compress the final stream. Decompression is computationally less expensive than compression. After Huffman decoding, only an inverse DCT has to be applied, and optionally the mid/sid stereo coding has to be converted back. For this reason, MP3 players can be easily realized as mobile devices and build into cell phones. The compression rates of MP3 encoders vary depending on the content, the chosen bitrate, and the quality of the encoder. 128kbit/s for music that comes from a stereo CD (raw 1.34Mbit/s) is not a-typical, thereby achieving a compression of about 1:10.

Perceptual Video Compression

The search for efficient video compression techniques dominated much of the multimedia research activity since the early 1980s. The first major milestone was the ITU H.261 encoder, from which JPEG later adopted the idea of quantizing DCT coefficients. Since then advancements have been made mostly in the field of motion estimation and with TV and cinema becoming digital, algorithms have been adopted into day-to-day use. Codecs have been mostly standardized by the ITU and ISO and popular standards are:

ISO/IEC 11172 aka MPEG 1 is designed to compress VHS-quality raw digital video and CD-quality audio down to 1.5 Mbit/s (26:1 and 6:1 compression ratios respectively). It is mostly used in video CDs, older digital cable and satellite TV.

ISO/IEC 13818 aka MPEG 2 is designed to be higher quality than MPEG1 while being backward compatible. It is used for encoding DVDs and digital TV of all kind (DVB-x).

ITU-T H.264 (formerly ISO/IEC 14496-10) aka MPEG4 is used for high-quality high-definition video, such as on Blu-Ray Discs or in the iTunes Store.

Since approximately 2000 the focus for multimedia research on video codecs as well as then MPEG standardization has been more on meta data and video search, resulting in MPEG-7 and MPEG-21.

In essence, video compressors are a combination of audio and image compressors that usually account for the fact that the images only differ slightly from frame to frame. The popular MPEG 1 and 2 video compression algorithms, for example, are conceptually a combination of JPEG, MP3, and the motion compensation technique, already described in Chapter XXX (differential coding).

Index Terms

Exercises

1. Implement the 2-dimensional Discrete Cosine Transform and the 2-dimensional Inverse Discrete Cosine Transform in a programming language of your choice. Apply your DFT and IDFT implementation to different signals of your choice and visualize them.

2. For a grayscale image implement a ``poor-man's JPEG" algorithm by applying your DCT implementation to the image, applying quantization, and then retransforming.
3. Which class of images is most/least prone to ugly JPEG artifacts? First think about it, then try different images in the implementation from exercise 2.
4. What effects would increasing the blocksize from 8x8 to 16x16, 32x32 or even higher have on the JPEG algorithm?
5. How would you parallelize the JPEG algorithm on a manycore processor?
6. What happens when you encode a JPEG using JPEG repeatedly? Explain.
7. Mix a sound file of your choice with different levels of noise. Explain the effects.
8. Write pseudo-code for the search for an optimal parameter configuration for the quantization of an MP3 granule given different bands, a quality function, and a minimum and maximum target bitrate. Analyze the runtime and discuss possibilities to optimize the runtime with and without trading of accuracy.

Literature

Paul S. Addison, *The Illustrated Wavelet Transform Handbook*, Institute of Physics, 2002, ISBN 0-7503-0692-0.
Ingrid Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, 1992, ISBN 0-89871-274-2.
Oppenheim, Alan V.; Schafer, Ronald W.; Buck, John A. (1999). *Discrete-time signal processing*. Upper Saddle River, N.J.: Prentice Hall. pp. 468–471.

Research Articles

N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform", *IEEE Trans. Computers*, 90-93, Jan 1974.
Brandenburg & Stoll, "ISO-MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio", J. Audio Eng. Soc 42 (1994) p 780-792.
Bosi et al. "ISO/IEC MPEG-2 AAC", J. Audio Eng. Soc. 45 (1997) p 789-814
S.A. Broughton, K. Bryan *Discrete Fourier Analysis and Wavelets: Applications to Signal and Image Processing* (2008 Wiley) p. 72
Carl Friedrich Gauss, 1866. "Nachlass: Theoria interpolationis methodo nova tractata," *Werke* band **3**, 265–327. Göttingen: Königliche Gesellschaft der Wissenschaften.
Harris, Fredric j. (January 1978). "On the use of Windows for Harmonic Analysis with the Discrete Fourier Transform". *Proceedings of the IEEE* **66** (1): 51–83. Article on FFT windows which introduced many of the key metrics used to compare windows.
Johnston and Ferreira, Sum-Difference Stereo Transform Coding, Proc. IEEE ICASSP (1992) p 569-571.
Moore, B.C.J. (1998) *Cochlear Hearing Loss*, London, Whurr Publishers Ltd
E. Terhardt, "Calculating virtual pitch", Hearing Res., vol. 1, pp. 155-182, 1979.
J. P. Princen and A. W. Johnson and A. B. Bradley, "Subband/transform coding using filter bank designs based on time domain aliasing cancellation," *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)* **12**, 2161-2164 (1987)

Chapter 16: Speech Compression

While the compression techniques presented so far have assumed generic acoustic or visual content, the following chapter presents lossy compression techniques that where especially designed for a particular type of acoustic data: Human speech. Almost every human being on earth talks virtually every day -- needless to say, there is a lot of captured digital speech content. Every movie or TV show contains and audio track, of which usually most of it is spoken language. The most important use of captured speech, however, is for communication, such as in cell-phones, voice-over-IP applications, or as part of video conferencing and meeting recordings. Most of the compression concepts discussed so far will also work on speech. The algorithms presented in the following though will achieve a higher compression ratio wile preserving higher perceptual quality by exploiting speech-specific properties of the audio signal. We already discussed what makes human speech different from generic audio content in Chapter XXX. We'll therefore assume that knowledge in this chapter and directly dig into the algorithmic part.

Properties of a Speech Coder

As explained already in Chapter XXX, the properties of every sound are defined by the properties of the objects that create the sounds, by the environment that the sound waves travel in, and by the characteristics of the receiver and/or capturing device. The object that creates human speech is the vocal tract. Vocal tracts also exist in animals, such as birds or cats. As we all know, the sounds they produce differ substantially from average human speech, so creating a bird-sing compression or cat's meouw encoding algorithm would also be substantially different. The following algorithms all try to exploit the characteristics of speech and have very limited applicability to music or other non-speech. However, all of them are of outmost importance to multimedia computing since they are used by millions of (mostly unaware) people in everyday life.

Speech codecs are applied in two main areas: Telephony and voice over IP applications. While the two areas seem to increasingly merge, historically the problems presented themselves in the areas required slightly different solutions. Other applications, such as Internet radio or speech compression for archival purposes yet introduce other priorities.

In general, the targets for a speech coder are:

- Good compression
- Minimum impact on speech intelligibility
- Maximum preservation of speaker characteristics

- Good handling of background noise (e.g. background noise should be eliminated when speech is present, but transmitted when not)
- Low computational complexity (e.g. cell phones needs to be small and should not need)
- Minimum latency, i.e. the delay introduced between transmitter and receiver must be small
- Transmission-error resistance (e.g. packet losses might be concealed)
- Text-independence

Sometimes, language-independence is sacrificed for a more efficient codec when a specific target market is aimed at.

Linear Predictive Coding (LPC)

One of the earliest models used for the compression of speech sounds is the so-called Linear Predictive Coding, or LPC. It is widely used in many places from telephony to military applications (low quality, ultra-low bandwidth) and is part of many, if not most, of todays speech compression algorithms in cell-phones. Typical rates of encoded streams vary from 800bits per second to 16kbits per second.

As the name implies, LPC is a predictive coder -- and it's linear. The underlying assumption of LPC is that a speech signal is produced by a buzzer at the end of a tube, with occasional added hissing and popping sounds (sibilants and plosive sounds). Although apparently crude, this model is a close approximation to the reality of speech production: The vocal tract forms a tube, which is characterized by its resonances due to its shape and the buzz is produced by the glottis, the space between the vocal folds, and is characterized by its intensity. Some literature calls methods that model signal production for efficient representation parametric redundancy exploitation methods. Figure 1 shows a diagram of one of the earliest LPC algorithms.

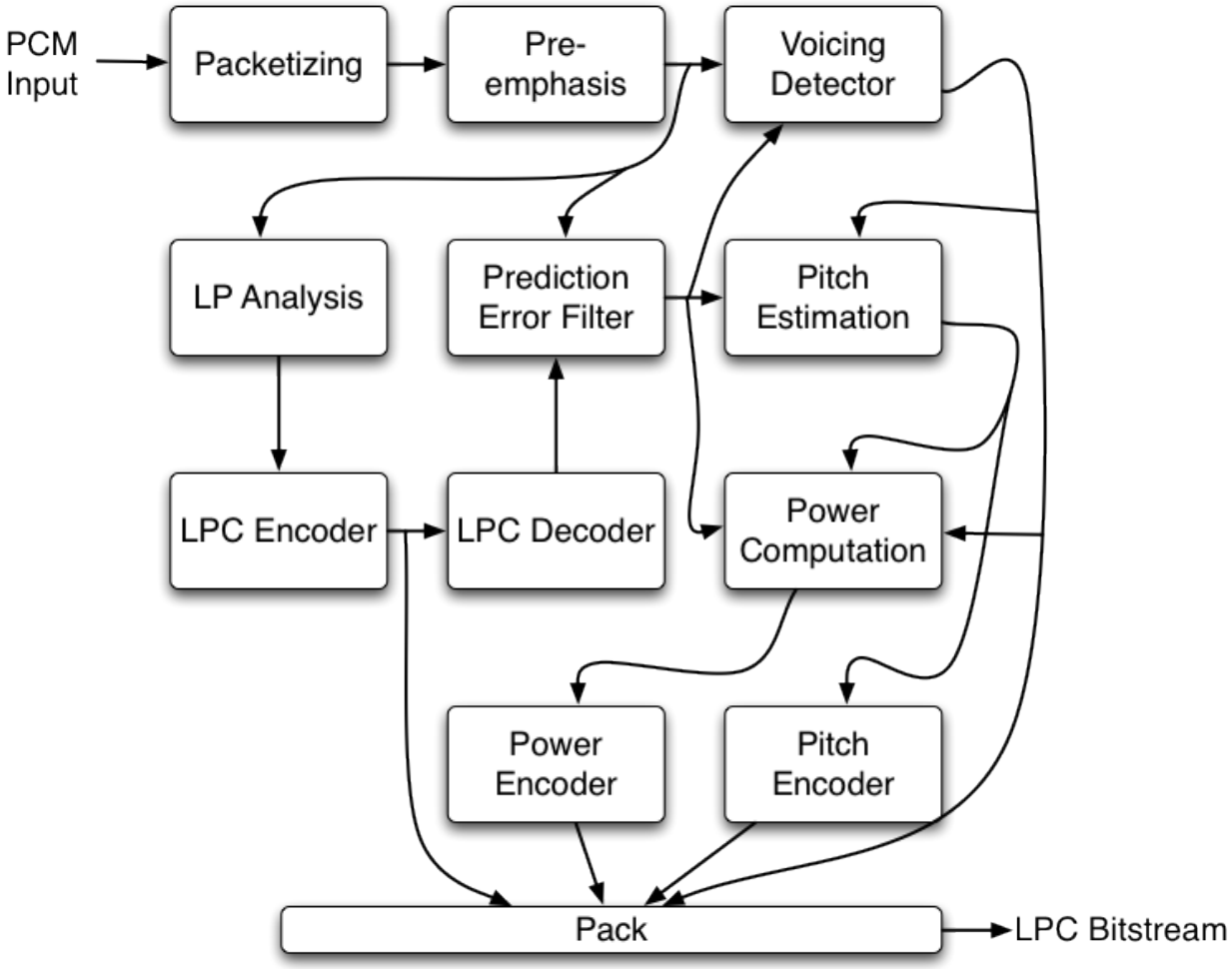


Figure 1. The LPC compression algorithm as defined in the NATO standard FS-1015 (LPC10)

It consists of several steps that are explained as follows. First, the sampled audio signal is packetized into small segments with a typical length of about 10ms. These atomic segments are usually called frames (compare video frames). A pre-emphasis filter increases the magnitude of the higher frequency parts with respect to the lower frequencies in order to increase the quality of the following steps. This trick is used often to increase the perceptual signal-to-noise ratio. This enhanced signal is then used for several steps. First, a voiced/unvoiced detector finds the regions where pitch can be calculated, eg. speech regions that contains mostly vowels. The pitch and the power are then also estimated and later encoded into the bitstream. In parallel, an LP analysis is performed on the signal. The LP signal is then used for error prediction as discussed in the previous chapter. The difference between the LP estimation and the actual signal is encoded together with the power and pitch in the actual bitstream. If the frame is voiced the pitch prediction is es-

timation from the prediction error signal. The compression achieved usually allows to transmit an 8kHz mono using 2400 bits per second in an understandable quality. The compression works in realtime even on home-sized computers from the 1970s. Not only is the LPC speech compression a very fundamental algorithm that is used in many different versions in a whole range of speech processing devices, it also allows us to discuss a couple of fundamental speech processing techniques.

Voiced/Unvoiced Detection

How can we determine if a speech frame contains a vowel or not? The methods currently available for doing this are not perfectly accurate but work in about 99% of the cases. Usually, a bag of features is used, combining the results from several computations on the signal. The most obvious feature is energy: In order to have pitch, this means in order to have periodicity, the signal must cross the zero amplitude line a couple of times. Thus the integral (or the sum of samples) of that signal should be close to zero. An unpitched signal can have any shape and the integral might be heavily biased towards a positive or negative number. To determine the energy E of a frame of length N containing samples s and ending at instant m the following simple equation is usually used:

Energy is always a positive number, hence the square root. Alternatively, the absolute value of the sample can be used, resulting in the so-called Magnitude-Sum Function:

As already said, a voiced signal might cross the zero amplitude line more often than a non-pitched signal. Of course, this can also be measured directly, by calculating the Zero-Crossing-Rate SC:

$$SC_{[m]} = \frac{1}{2} \sum_{n=m-N+1}^{m} \left| \mathrm{sgn}(s_{[n]}) - \mathrm{sgn}\, s_{[n-1]}) \right|$$

The *sgn()* function returns 1 or 0 depending on the sign of the operand. A third method is to calculate the prediction gain:

$$PG_{[m]} = 10\log_{10}\left(\frac{\sum_{n=m-N+1}^{m} s^2{}_{[n]}}{\sum_{n=m-N+1}^{m} e^2{}_{[n]}}\right)$$

It can be observed that voiced frames on average achieve 3 dB or more in LPC prediction gain than unvoiced frames, mainly due to the fact that periodicity implies higher correlation among samples, and thus easier to predict. Unvoiced frames, are more random and therefore less predictable. For very low-amplitude frames, prediction gain is normally not calculated to avoid numerical problems; in this case, the frame can be assigned as unvoiced just by verifying the energy level. Thresholding energy, zero-crossing rate and prediction gain is not an exact science. Modern systems use machine learning to find good values for estimating these values on a concrete data set. Finding a perfect boundary between a voiced and an unvoiced segment is nearly impossible.

Pitch Estimation

The fundamental frequency (F0) of a periodic signal is the inverse of its period. The subjective "pitch" of a sound usually depends on its fundamental frequency but also depends on other factors. Given that a frame is voiced, estimating the pitch of the frame, however, is one of the most important and frequently demanded operations in audio processing. In speech processing, the pitch period is defined as the time between successive vocal cord openings. It is important to note that expected values for the for men, possible pitch periods lie between 4ms and 20ms (frequency between 50Hz and 250Hz) and for women and children between about 2ms to 8ms (frequency between 120Hz and 500Hz). Unfortunately, estimating this time is, like the voiced/unvoiced detection, not an exact science either. Unless the signal is artificially generated, pitch period estimation is a complex undertaking due to the lack of perfect periodicity in real world signals. Unless trained thoroughly, even a singer's voice has no perfect pitch due to interference with formants of the vocal tract (voice impurities). Also, the uncertainty of the starting point of a voiced segment (see previous paragraph) and other real world problems, such as noise and echo make perfect pitch estimation hard. For this reason, arbitrarily complex pitch estimation algorithms have been developed and refinements will still be presented in research papers to come. In practice, pitch period estimation is implemented as a trade-off between computational complexity and performance. From the many algorithms that have been proposed for this task, only two will be presented here. The first, and most frequent method, is the so-called autocorrelation method.

The autocorrelation value reflects the similarity between the frame *s[n]* and the time-shifted version *s[n-l]*. Again, the frame has length *N* containing and is ending at instant *m*. The variable *l* is a positive integer representing a time lag and *n= [m–N+1,m]*. The range of lag is selected so that it covers a wide range of pitch period values. For example, at 8kHz sampling rate, if *l* is between 20 and 147 (2.5 ms to 18.3 ms), the possible pitch estimation times values range from 54.4 Hz to 400 Hz. By calculating the autocorrelation values for the entire range of lag, it is possible to find the value of *l* associated with the highest autocorrelation representing the pitch period estimate. In other words, the autocorrelation *R* is maximized when the lag *l* is equal to the pitch period.

The pseudo-code for the algorithm would look like this:

```
// Input: last index in frame m, number of samples N,
// sampling rate sr per second
// Output: The pitch period in seconds.
pitch(m, N, sr)
    peak := 0
    FOR l:=20 TO 150
        autoc:=0
        FOR n:=m-N+1 TO m
         autoc:=autoc+s[n]*s[n-l]
    IF (autoc>peak)
        peak:=autoc
        lag:=l
pitch := lag/sr
```

A second method for pitch estimation is the so-called Magnitude-Difference Function. The idea is that for short segments of voiced speech it is reasonable to expect that s[n]-s[n-l] is small for l =0,±T,±2T, . . , with T being the signal's period. By computing the MDF for the lag l range of interest, we can estimate the period by locating the lag value associated with the minimum magnitude difference. And here is the equation for the MDF:

$$MDF[l,m] = \sum_{n=m-N+1}^{m} \left| S[n] - S[n-l] \right|$$

Note that from the same equation, each additional accumulation of term causes the result to be greater than or equal to the previous sum since each term is positive. Thus, it is not necessary to calculate the sum entirely: if the accumulated result at any instance during the iteration loop is greater than the minimum found so far, calculation stops and resumes with the next lag. Also, no multiplication is involved in this method which is sometimes interesting for speed and memory usage on small devices. Overall, this method is faster than the regular autocorrelation method. The idea is implemented with the following pseudocode:

```
// Input: last index in frame m, number of samples N, sampling rate sr per
second
// Output: The pitch period in seconds.
pitch2(m, N, sr)
    min := infinity
    FOR l:=20 TO 150
        mdf:=0
        FOR n:=m-N+1 TO m
            mdf:=mdf+ABS(s[n]-s[n-l])
            IF (mdf>=min) BREAK
        IF (mdf<min)
            min:=mdf
            lag:=l
pitch2 := lag/sr
```

It is important to note here, that the two methods present really only estimate pitch. In fact, nobody will probably ever be able to accurately calculate pitch, since pitch, even though depending on the fundamental frequency, is subjective. Even when focussing only on speech, there are many factors that distort pitch. For example, periodic vibration at the glottis may produce speech that is less perfectly periodic because of movements of the vocal tract that filters the glottal source waveform. Then, glottal vibration itself may also show aperiodicities, such as changes in amplitude, rate, or glottal waveform shape. Reverberation inside the vocal tract also distort pitch.

LP Analysis

The underlying model for LPC is this:

$$\tilde{x}(n) = a_1 x(n-1) + a_2 x(n-2) + \cdots + a_M x(n-M) = \sum_{i=1}^{M} a_i x(n-i)$$

where x~(n) is the prediction of the present sample generated through linear combination of the past M samples x(n) to x(n-i). The a_i are called the linear prediction coefficients. In other words the predictor tries to predict a sample as a linear combination of the previous outputs. Usually 10-32 linear prediction coefficients are used. The number of coefficients is usually chosen depending on the frequency used, for 8kHz sampling rate 10-dimensional LPC analysis is usually good, for 16kHz, 20 is a typical value.

The idea is then to minimize the error between the actual and the predicted value:

$$\varepsilon(n) = x(n) - \tilde{x}(n) = x(n) - \sum_{i=1}^{M} a_i x(n-i)\,.$$

This is usually solved by mathematical optimization, such as Levinson-Durbin recursive method, the scientific literature also discusses other methods (see research references). For an order N filter, the filter coefficients a_i are found by solving the $N \times N$ linear system $Ra=r$, where

$$\mathbf{R} = \begin{bmatrix} R(0) & R(1) & \cdots & R(N-1) \\ R(1) & R(0) & \cdots & R(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(N-1) & R(N-2) & \cdots & R(0) \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(N) \end{bmatrix}$$

with $R(m)$ being the auto-correlation of the signal $x[n]$, computed as described above (and implemented below):

$$R(m) = \sum_{i=0}^{N-1} x[i]x[i-m]$$

and r being the so-called reflection coefficient. The following pseudo-code illustrates a practical implementation of this method which was first invented by N. Levinson in 1947 and then modified by J. Durbin in 1959 (see research references).

```
// Compute LPC coefficients from a series of auto-correlation coefficients
// Input:  dim order of LPC analysis,
//         ac [0...dim+1]autocorrelation values (see helper function below)
// Output: ref[0...dim] reflection coefficients R(N),
//         lpc[0...dim] LPC coefficients,
//         error        residual error
levinson_durbin(dim, ac)
      error := ac[0]
      IF (error == 0)
            set all ref[i]:=0
            levinson_durbin := ([0...0],[0...0],0)
            // return and exit routine
      // main loop
      FOR i:=0 TO dim-1
            // Calculate the reflection coefficient
            r:=-ac[i+1]
            FOR j:=0 TO i
                  r:=r-lpc[j]*ac[i-j]
                  r:=r/error
                  ref[i]:=r
```

```
            // Update LPC coefficients and total error
            lpc[i]:=r
            FOR j:= 0 TO (i/2)-1
                  temp := lpc[j]
                  lpc[j] := lpc[j]+r*lpc[i-1-j]
                  lpc[i-1-j] := lpc[i-1-j]+r*temp
            IF (i%2 == 1)
                  lpc[j] := lpc[j]+lpc[j]*r
            error := 1.0-r*r
levinson_durbin := (ref,lpc,error)
```

The autocorrelation values can be calculated as follows:

```
// Compute the autocorrelation coefficients needed for the LPC
// Input:  n number of audio samples,
//         x[0..n-1] audio samples
//         lag a maximum lag range
// Output: ac[0...lag-1] autocorrelation values
lpc_autocorrelation(n, x, lag)
      WHILE (lag>0)
            lag:=lag-1
            FOR i:=lag TO n
                  d:=0
                  d:=d+x[i]*x[i-lag]
            ac[lag]:=d
lpc_autocorrelation:=ac
```

With the increasing availability of multiple CPU core architectures, an alternative solution to the Levinson-Durbin recursion is in frequent use that is easier to parallelize. The so-called Schuer recursion is related to the Levinson-Durbin method but faster on parallel architectures. On multiple cores, Levinson-Durbin would take time proportional to *O(dim*log(dim))*, Schuer requires time proportional to *dim*. The following pseudo code, which again relies on *lpc_autocorrelation*, shows the idea:

```
// Alternative recursion algorithm for parallel architectures
// Input:  dim order of LPC analysis,
//         ac [0...dim+1]autocorrelation values (see helper function below)
// Output: ref[0...dim] reflection coefficients R(N),
//         error        residual error

schuer(dim,ac)
      error := ac[0]
      IF (error == 0)
            set all ref[i]:=0
            schur := ([0...0],0)

      // Create a so-called generator matrix G with dimensions [2,dim]
      FOR i := 0 TO DIM-1
            //  Calculate this iteration's reflection coefficient and error.
            G[0][i] := ac[i+1]
```

```
            G[1][i] := ac[i+1]
      i:=0
      WHILE (true)
            r := -G[1][0]/error
            ref[i] := r
            error := error + G[1][0]*r
            i:=i+1
            IF (i>=dim)
                  schur := (ref,error)
                  // return and exit routine
            // Update the generator matrix.
            // Unlike the Levinson-Durbin summing of reflection coefficients,

            // this loop could be distributed to many processors which
            // each take only constant time.
            FOR m := 0 TO dim-i-1
                  G[1][m] := G[1][m+1] + r * G[0][m]
                  G[0][m] := G[1][m+1] * r + G[0][m]
schur := (ref,error)
```

The calculation of the related LPC coefficients is left as an exercise.

As mentioned above, the building blocks introduced here for explaining the LPC algorithm have been reused many times in other speech compression algorithms. LPC modeling is also used for speech synthesis. For example, the very popular "speak'n'spell" toy from the 1980s (see web references) has used LPC for reading words. LPC can also be used as a feature in speech analysis, eg. for speech or speaker recognition. Most importantly though, the LPC algorithm was used as a basis for further, more complex algorithms for speech compression, of which some are described in the next sections.

CELP

CELP is an enhancement of the LPC compression providing better quality speech than LPC-10e, described above, without increasing the bit rate too much. Since LPC is a synthesizer, a natural extension is to use a wavetable (see Chapter XXX) to increase the quality, trading off the increase of coder and decoder complexity for smaller bitrates. Therefore, many successful methods use a so-called codebook, usually a table of typical residue signals. In a nutshell, the analyzer compares the residue to all the entries in the codebook, chooses the entry which is the closest match, and just sends a reference to that entry. The synthesizer receives the reference, retrieves the corresponding residue from the codebook, and uses the "code" to "excite" the re-synthezised signal, hence the name Code Excited Linear Prediction (CELP). The principle behind CELP is also called analysis by synthesis because the encoding (analysis) is performed by perceptually optimizing the decoded (synthesis) signal in a closed loop.

CELP search is broken down into several steps. Typically, encoding is performed in the following order:

1. Linear prediction coefficients are computed, converted to Line Spectral Frequencies (see below), and then quantized
2. An adaptive codebook is searched and its contribution removed
3. A fixed codebook is searched

The steps will be explained in the following.

LSF encoding

An essential trick is to transform the linear prediction coefficients into so-called line spectral frequencies (LSF), sometimes also called line spectral pairs (LSP).

The linear prediction polynomial (from above) can also be written as

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}$$

which can be decomposed into the following two complex equations:

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1})$$
$$Q(z) = A(z) - z^{-(p+1)}A(z^{-1})$$

where *P(z)* is said to correspond to the vocal tract with the glottis closed and *Q(z)* with the glottis open. The reason for this transformation is to exploit the following mathematical trick.

A(z) has complex roots anywhere within the unit circle (z-transform) but *P(z)* and *Q(z)* have the very useful property of only having roots directly on the unit circle. So in order to find the roots, one takes a test point $z = exp(jw)$ and evaluates $P(exp(jw))$ and $Q(exp(jw))$ using a grid of points between 0 and π. The zeros of P(z) and Q(z) also happen to be interspersed which is why one swaps coefficients as one find roots. Therefore the process of finding the LSP frequencies is finding the roots of two polynomials of order $p+1$.

And here is why this helps: LPC coefficients do not quantize well since small quantization errors may lead to large spectral distortion. Of course, the higher order bits in the representation of the coefficients are naturally more sensitive to transmission errors than the lower-order ones. Also the LPC coefficients do not interpolate well, i.e. one cannot compute them at two distinct times

and expect to accurately predict values in between. Therefore, instead of encoding the coefficients, it would be better to encode the zeros of the LPC equation. However, finding these zeros numerically entails a computationally complex two-dimensional search, while the zeros of $P(x)$ and $Q(x)$ can be found by simple one-dimensional search techniques. Over the years, it has been found that LSP frequencies quantize well and interpolate better than all other parameters that have been tried in speech applications.

To convert back to LPCs, one evaluates $A(z) = 0.5[P(z) + Q(z)]$ by putting signal samples through it the order of the LPC times, yielding back the original $A(z)$.

Adaptive Codebook

The entries in the adaptive codebook consist of delayed versions of the excitation to make it possible to efficiently code the portions of the signal that are periodic, such as the voiced parts of speech. In the decoder, the final excitation is produced by summing the contributions from the adaptive codebook and the fixed codebook (see below):

$$e[n] = e_a[n] + e_f[n]$$

where $e_a[n]$ is the adaptive codebook contribution and $e_f[n]$ is the fixed codebook contribution. The filter that shapes the excitation has an all-pole model of the form $1/A(z)$, where $A(z)$ are obtained using linear prediction. An all-pole filter is used because it is a good representation of the human vocal tract and because it is easy to compute.

Fixed Codebook

A good choice for a fixed codebook is normal distributed random vectors. The reason for this is that the LPC-filtering and the adaptive codebook already removes large parts of the interdependencies between the samples yielding relatively white-noise-like residual. Usually, the codebook comprises of about 1024 excitation vectors (10-bit codebook) for a 40-sample subframe with 8 kHz sampling frequency.

The methods for finding the right entry in the adaptive codebook vary from coder to coder. Many coders simply uses the Euclidian distance to determine the similarity between code vectors. Speex uses the Euclidian distance shaped with a perceptual weighting functions that tries to roughly approximate noise perception in the human ear. Figure 2 shows the optimization loop.

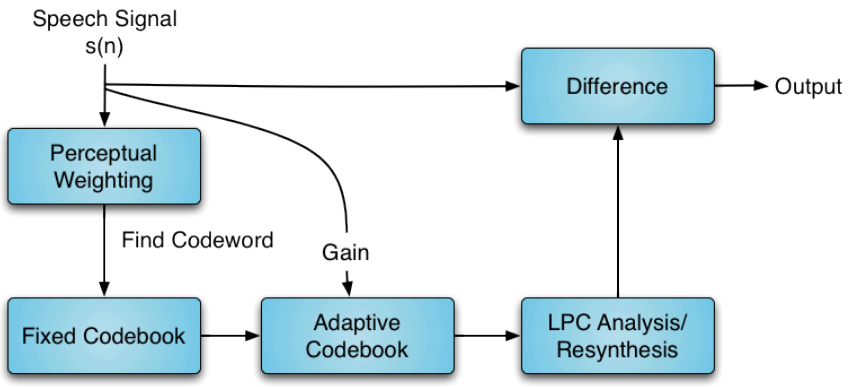


Figure 2. The CELP encoder loop performing analysis by synthesis until the perceptually weighted difference is minimal.

In the end, the encoder transmits 144 bits of information based on 240 audio samples of speech (30 ms). The bit allocation is shown in Table 1.

Parameters	No. of bits
LSF	34
Adaptive Filter	48
Fixed Codebook Index	36
Gains	20
Synchronization	1
Error Correction	4
Future Use	4
Total	**144**

Table 1. A typical CELP bitstream.

The CELP algorithm is most prominently described in the NATO standard FS 1016 which provides good quality, natural sounding speech at 4800 bit/s and in ITU-T recommendation G.728 operating at 16 kbit/s. The popular open source speech codec Speex (see web references) is also based on CELP. Speex is targeted at voice over IP applications.

GSM

GSM stands for Global System for Mobile communications and was originally developed by a group called Groupe Spécial Mobile with the same acronym. The group was formed by the Conference of European Posts and Telegraphs (CEPT) in 1982 in an effort to develop a pan-European public land mobile system. Today, GSM is the number one standard for mobile phones in the world. The GSM Association, estimates that 80% of the global mobile market uses the standard. Its ubiquity, which now spans the world, makes international roaming easily possible and enables subscribers to use their phones in many parts of the world. The main difference between GSM and its predecessors in that both signaling and speech channels are digital, and thus is considered a second generation (2G) mobile phone system. This also makes data transmission over the same line very easy, enabling services such as text and multimedia messaging. The GSM standard describes more than just voice compression. It specifies everything necessary to build a global communication infrastructure, such as the structure of the radio network, the frequency ranges, the antennas and cells, subscriber identification mechanisms, security standards, power control, and so on. For further information refer to the references and Chapter XXX. This section only provides a brief overview of the speech codecs defined in GSM, which are mainly based on LPC.

GSM has used a variety of voice codecs to compress 3.1 kHz sampling rate audio captured by the cell phone into between 6.5 and 13 kbit/s. In the original specification, only a so-called Half Rate (5.6 kbit/s) and Full Rate (13 kbit/s) codec were defines. These used a system based on linear predictive coding.

In 1997, the Enhanced Full Rate (EFR) codec working on a 12.2 kbit/s basis was introduced. With the development of UMTS, this codec was modified into the so-called AMR (Adaptive Multi-Rate) codecs. This is a variable-rate codec which is high quality and robust against interference when used on Full Rate channels, and less robust but still relatively high quality when used in good radio conditions on Half Rate channels.

Even though still built from almost exclusively the concepts explained in this chapter, current codecs, such as GSM-AMR have grown in complexity beyond a size that can be presented in pseudo-code in this book. We therefore limit ourselves to presenting and discussing the block diagram, shown in Figure 3.

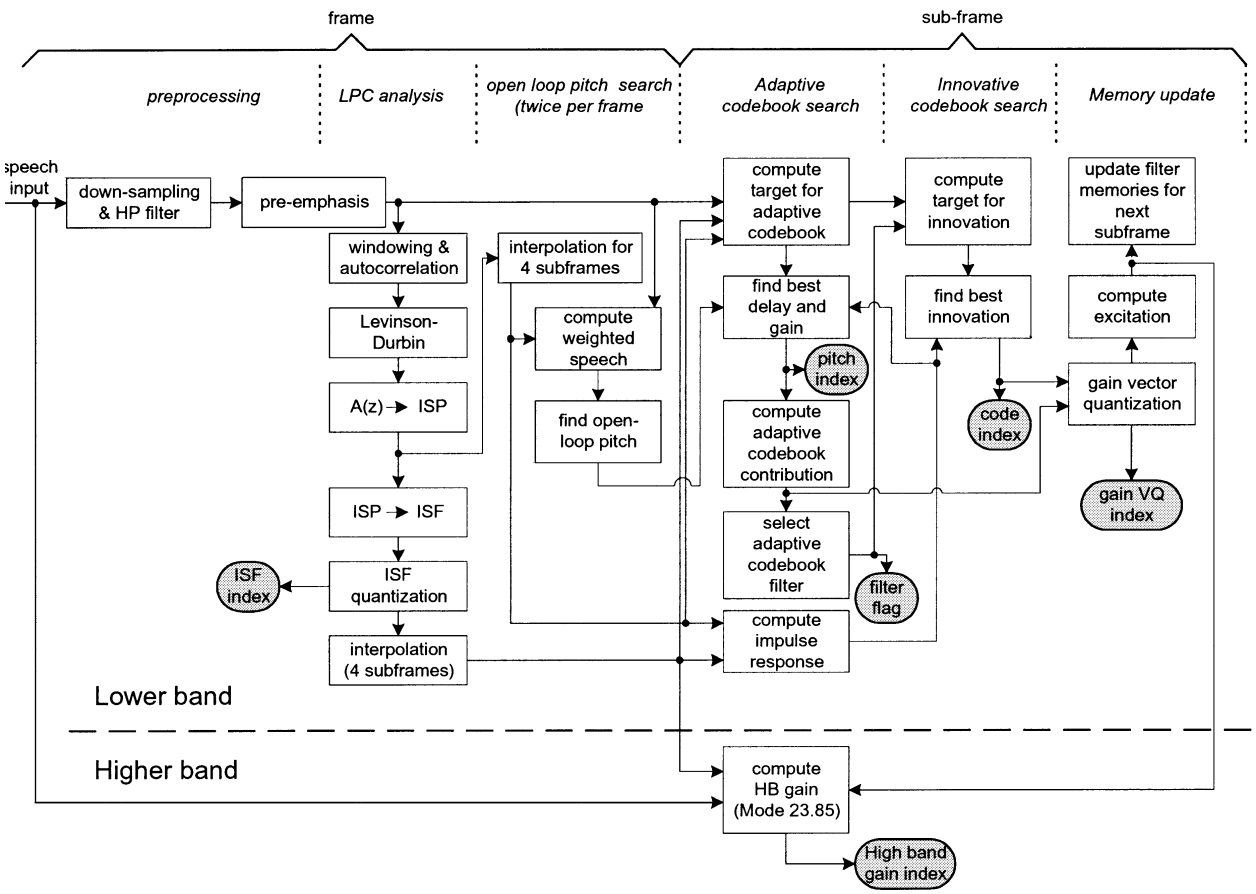


Figure 3. The structure of the GSM-AMR used in millions of cell-phones today (diagram by Bessette et al.).

The GSM AMR-WB codec is based on a special version of the CELP codec, the so-called Algebraic Code Excited Linear Prediction (ACELP) algorithm. ACELP is uses an algebraic adaptive codebook, i.e. the codebook entries are not stored explicitly but as mathematical formulas. The main advantage is that the codebook it uses can be made very large (> 50 bits) without running into storage (RAM/ROM) or complexity (CPU time) problems. The main disadvantage is that the technology is patented and is therefore not freely available. Although the ACELP coder gives very good performance on narrow-band signals, some difficulties arise when applying the telephone-band optimized ACELP model to wideband speech, therefore additional features needed to be added to the model for obtaining high quality on wideband signals. The GSM-AMR codec works in different modes for different bandwidth. The bitrates are 23.85, 23.05, 19.85, 18.25, 15.85, 14.25, 12.65, 8.85 and 6.6 kb/s.

The input signal is down-sampled and pre-processed using a high-pass filter and a noise-reduction filter. The ACELP algorithm is then applied to the down-sampled and pre-processed

signal. Linear Prediction analysis is then performed once per 20 ms frame. The set of linear prediction parameters is converted to, so-called immittance spectrum pairs (ISP) (see research references), which is a different form of LSF, and quantized using vector quantization (VQ). The speech frame is divided into subframes. The adaptive and fixed codebook parameters are transmitted every subframe. The pitch lag is encoded with 9 bits in odd subframes and relatively encoded with 6 bits in even subframes. Another bit per subframe is used to determine the low pass filter applied to the past excitation. The pitch and algebraic codebook gains are jointly quantized using 7 bits per subframe.

The highest frequency band (6400–7000 Hz) is reconstructed in the decoder using the parameters of the lower band and a random excitation. No information about the higher band is transmitted, except in the best mode (23.85kb/s), where the higher band gain is transmitted using 4 bits per subframe. In other modes, the gain of the higher band is adjusted relative to the lower band using voicing information. The spectrum of the higher band is reconstructed by using a wideband LP filter generated from the lower band LP filter.

As a result, the decoder has a similar structure, as shown in Figure 4.

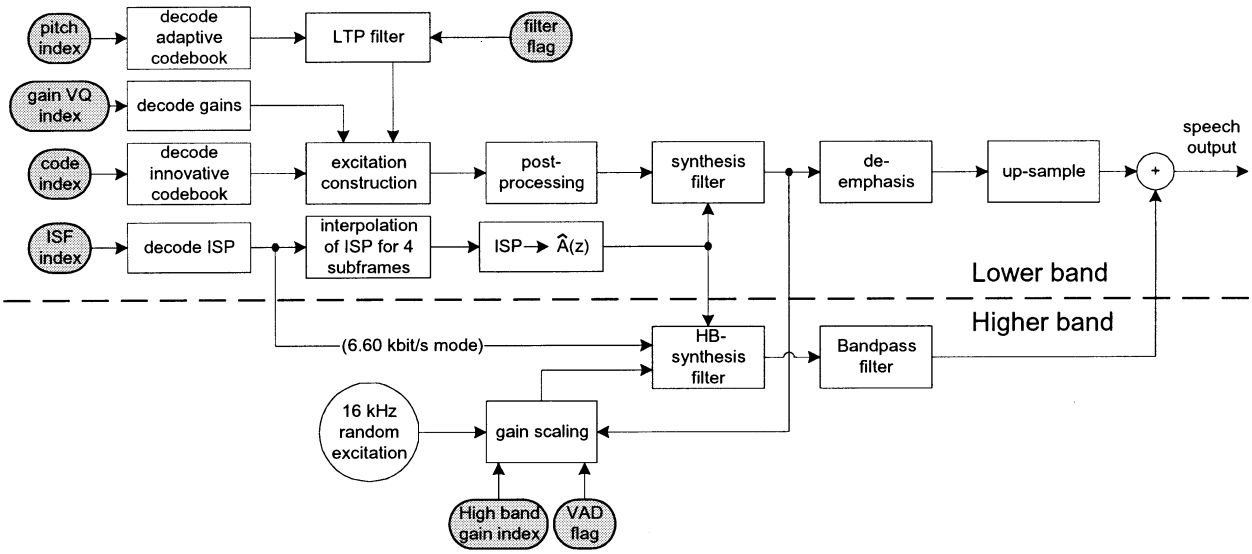


Figure 4. The structure of the GSM-AMR used in millions of cell-phones today (diagram by Bessette et al).

Not only mobile phones use LPC variants, Skype's SVOPC and the later SILK codec use them too. SVOPC and Silk are also tuned to conceal frame drops. This is done by interpolating LSF parameters between two received frames to make up for the missing one.

All speech codecs have in common that they use a model of speech production to save bits. The next chapter will explain perceptual codecs. These use a model of perception. Of course, these codecs can be used for speech compression as well, although, as of today, speech-production coders still seem to do better on speech than perceptual coders.

Index Terms

Exercises

1. Elaborate which target properties for a speech coder would be most important for: A regular phone, a cell phone, Internet radio, and an Internet teleconferencing application.
2. Describe how speech intelligibility differs from audio quality.
3. Implement a voiced/unvoiced detector. Then try it on different voice recordings and describe the limits of your approach
4. Implement a pitch estimator. Then try it on different voice and music recordings and describe the limits of the approach.
5. Perform a runtime-analysis of the Levins-Durbin and Schuer algorithms on a single CPU. Then describe how the Schuer algorithm performs better on multiple CPUs. Calculate the run-time for it on different CPUs.
6. If one where to use LPC to model music, which types of instruments would be modeled well and which ones would not? Give examples and explain why.
7. Write the (pseudo-)code for calculating the LPC coefficients from the reflection coefficients as output from the Schuer pseudo-code presented in this chapter.
8. Write (pseudo-)code for calculating the LSF coefficients from LPC coefficients.
9. Write (pseudo-)code to implement a CELP encoder.
10. Use a current implementation of a speech encoder (e.g. in your cell phone or using a voice-over-IP application) and transmit speech, music, and noise through it. Describe and explain the artifacts observed.
11. Given the models presented in this chapter, discuss what other elements are contained in speech that were not discussed. Describe how these could be handled.
12. Use a speech compression algorithm of your choice and describe how you would handle packet loss?

Literature

P. Vary, R. Martin: Digital Speech Transmission: Enhancement, Coding and Error Concealment, Wiley, 2006

Web Links

Speak'n'spell history page: http://www.speaknspell.co.uk/
Speex: http://www.speex.org/
GMS World: http://www.gsmworld.com
Comp.Speech FAQ: http://www.speech.cs.cmu.edu/comp.speech/

Research Papers

- Levinson, N. (1947). "The Wiener RMS error criterion in filter design and prediction." *J. Math. Phys.*, v. 25, pp. 261-278.
- Durbin, J. (1960). "The fitting of time series models." *Rev. Inst. Int. Stat.*, v. 28, pp. 233-243.
- Trench, W. F. (1964). "An algorithm for the inversion of finite Toeplitz matrices." *J. Soc. Indust. Appl. Math.*, v. 12, pp. 515-522.
- Bessette, B. and Salami, R. and Lefebvre, R. and Jelinek, M. and Rotola-Pukkila, J. and Vainio, J. and Mikkola, H. and Jarvinen, K. : "The adaptive multirate wideband speech codec (AMR-WB)", IEEE Transactions on Speech and Audio Processing, Vol. 10, No. 8, pp. 620-636, 2002.
- Bistritz, Y. Peller, S.: "Immittance spectral pairs (ISP) for speech encoding", Proceedings of IEEE ICASSP, page(s): 9-12, April, 1993.
- B.S. Atal, "The History of Linear Prediction," *IEEE Signal Processing Magazine*, vol. 23, no. 2, March 2006, pp. 154–161.
- M. R. Schroeder and B. S. Atal, "Code-excited linear prediction (CELP): high-quality speech at very low bit rates," in *Proceedings of the IEEE ICASSP*, vol. 10, pp. 937–940, 1985.
- Xianglin, Wang; C.-C. Jay Kuo (May 1998). "An 800 bps VQ-based LPC voice coder". *The Journal of the Acoustical Society of America* **103** (5): 2778.

PART IV:
ORGANIZATION AND ANALYSIS OF MULTIMEDIA CONTENT

Chapter 17: Multimedia Information Retrieval

From all the topics that pertain to the analysis of multimedia data, Multimedia Information Retrieval (MIR) has recently and rapidly emerged as the most important technology needed for many questions faced by people in different aspects of their regular activities. Therefore, while not true for the entire field the next chapters will focus on multimedia organization and analysis, mostly from retrieval aspects. We begin by defining multimedia retrieval and the set of challenges and algorithms that dominate the field

Serious efforts in Multimedia Information Retrieval began in the early 1990s. Devices like digital cameras and phone cameras combined with progress in compression and availability of bandwidth brought a major change in the lifestyles of people first in advanced countries and then everywhere in the world. In fact, the rapid progress in technology created strong demand for organization and access to multimedia data, but the techniques for multimedia information retrieval have been slower to develop than the volume of data. The basic problem in MIR system is connecting different types of data sources to users with diverse background and different needs, as shown in Figure 1.

Multimedia computing addresses a problem that many other fields like computer vision, databases, and information retrieval face: connecting data and users. As shown in figure 1, data exists in many forms, ranging from bits to alphanumeric documents to photos and video. On the other hand users of the data in a modern computing environment may come from many different education backgrounds, of different culture, and of different socio-economic status. The challenge is how to connect a user with a data source so the user can use the data he needs to solve his application. A user is never interested in what and where the data is; she is only interested in solving the problem at hand. The major hurdle in connecting users to the data is often referred to as the semantic gap. This is explained in more detail in the Chapter on Context.

In this chapter, we will present basic concepts and techniques related to accessing multimedia data. We will start with the structured data in databases, discuss information retrieval to deal with accessing information in text, and then present techniques developed and being explored in MIR.

Multimedia Information Retrieval (MIR) contains three important components: Multimedia, Information, and Retrieval. Here are some general observations on the meaning of each of these terms in this context:

Multimedia: When thinking of the MIR problem, it is natural to think that increase in number of types of data such as images, text, and audio will result in increased complexity of organization, indexing, and retrieval, however, by adopting the right perspective and using opportunistic information from disparate sources, the availability of correlated and complementary multimedia data simplifies the problems significantly.

Information: MIR is naturally influenced by IR (Information Retrieval) which deals with text. In text most successes are because the information retrieved is directly available in the data. The information of interest in sensory data must be extracted by processing. Whenever IR systems try to retrieve information not directly available in data, they also face challenging problems, especially the so-called semantic gap, which we will discuss in section X

Retrieval is usually interpreted as the operation of accessing information from human or computer memory. Retrieval, query, and search are used to mean similar things, although there are subtle differences. All of these terms are related to finding appropriate information in some application context from a large volume of data in memory. Depending on the sources of data and the application context, sometimes one is interested in precise answers that are directly available in the data; sometimes in information that is derived from the data; and in other cases just finding related sources that may contain information. These cases have different scope and require different techniques. In most applications, retrieval is one step in the overall solution and the application context influences the requirements from this step significantly.

Today, even the WWW is dominated by text. Photos and videos are increasing on the Web, but are organized and usually accessed on WWW using tags and keywords. As discussed before, humans are extremely adept in dealing with sensory and symbolic information by effortlessly converting sensory information to symbolic form and processing this hybrid form of information effortlessly. Converting sensory data to symbols in computer systems has attracted significant research, but so far, has proven exceedingly difficult. A primary difficulty in developing computational techniques for automated sensory interpretation lies in our inability to formally represent and effectively model the appropriate context within which the sensory information should be interpreted. A specific impediment in using contextual information has been in our inability to constrain the scope of the potentially infinite number of uncertain and imprecise context-defining variables. To organize all multimedia experiences and making them as searchable using experiences as modern search engines have done using keywords, tremendous progress will be made in inductive reasoning to generalize and create new knowledge, and abductive reasoning to verify facts from data.

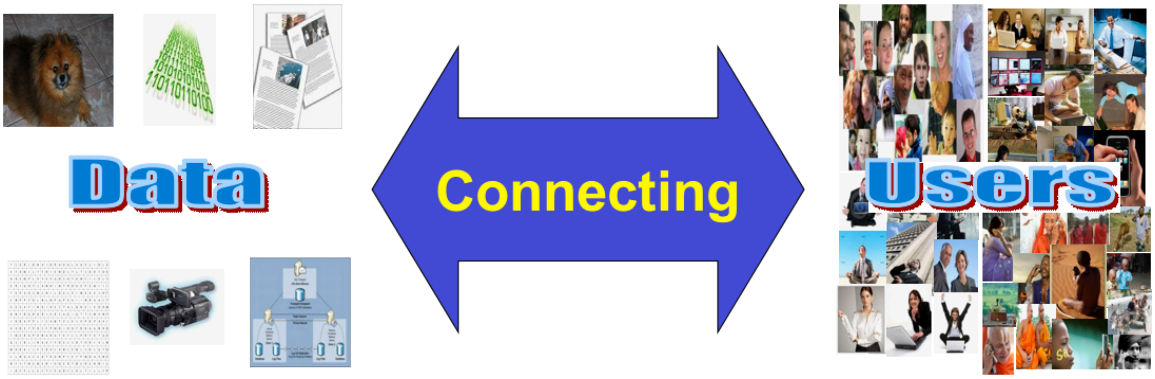


Figure 1: Multimedia data could be in any form ranging from text to signals like video. People from very diverse background may come to a repository in a MIR system to access the data to satisfy their needs.

Image Search on the Web

Most images on the Web early on appeared as parts of documents. In these cases the images could be considered secondary to the text material because images were used more to provide experiential component in an otherwise text document. Searching for images on the Web started out very similar to text search. Images are searched on the Web using keywords. The keywords that characterize images are usually from the following:

- Name of the image file,
- Text on the page where the image is located, or
- Tags assigned to the image.

In early systems for Web Image Search, image files were not even opened to analyze them. An image file was detected from the extension in the file name and then the search was performed using text. This approach does have limitations, but fits well with techniques used in text search. We will therefore begin this chapter by briefly describing text information retrieval.

In the last decade, many applications emerged where photos or images are the main part of the document. In these applications, there could be text that is used as the supporting material, but the photos are considered the main carrier of information and experiences. In such applications, techniques based on text analysis and keywords may be inadequate. In some of such applications, tags have been used, but as we will discuss, the limitations of tags make them only partially useful, and other techniques, more focused on image content, are being explored.

Structured and Un-structured data

The nature of data plays a very important role in how it can be organized and retrieved. Data may be in a structured format such that the sequence of data items is well defined and is known to the system. In such a case, system knows how to interpret data and can store it using indexing techniques for efficient retrieval of the data. At the other extreme is the case where all data appears in a form that is not known to the system and may be of random type or order. For such unstructured data, the system cannot use any indexing schemes and usually stores the data based on the name of the file and some other meta data such as the date of creation of the file containing the data.

Since there was a clear need to access an increasing volume of text data, commonly considered unstructured data, people developed techniques that could help in providing some structure to this new common type of data and introduced a new approach structure using Extensible Markup Language (XML). . XML uses a different approach to structuring data by asking document creators to introduce enough clues, or structure, in the document so that an automatic process can read what the document or a section of it is about. This metadata approach enables advanced systems to know more about the document than today's automatic techniques can. It also has the ability to work gracefully with more automation.

XML introduces structure in otherwise unstructured documents. That is, it structuralizes text. Multimedia data, like other data, must be stored using organization principles that will help enable management and retrieval. Moreover, multimedia data should be organized more carefully because of its time–serial nature and its enormous size. Another difficulty is that current metadata for audio, video, images, and other similar sources is more about the data than about its semantic content. The tags in XML introduce semantic partitioning of text. Techniques for introducing the semantic partitioning of video, audio, and images are needed. Multimedia researchers have spent considerable effort on developing automatic techniques for video and audio segmentation and for indexing images based on some basic characteristics such as color and texture. These techniques are very useful and will revolutionize how we'll organize multimedia data someday in the future. However, we need to organize multimedia data today. The current automatic techniques for semantic partitioning are even more infantile than those for text. The only solution may be to develop powerful approaches for structuralizing multimedia data, which could prove to be as revolutionary as the introduction of XML.

One can not provide organization and access to large volume of data without using some structure in the data. In some cases, such as relational databases, the data is created and made

available to the system in a well-structured format. In other cases when the data is not directly entered in the system, some techniques must be used to identify the structure that will help in providing the functionality for organizing and accessing the data. Multimedia information retrieval techniques, discussed in this chapter, are all about defining structure that must be used to organize and access the data and applying techniques for data analysis that will help in extracting this information from the data and storing it.

Databases

Searching for data is an old problem. When computers started becoming popular, and started finding applications in businesses as well as in our applications involving large volumes of data, it became important to develop systems that will help in organization, storage, management, and retrieval of data (OSMR). After early navigational and network databases, the relational model introduced by Edgar Codd became popular and became the foundation for most of the commercial database systems. The uniform set-theoretic representation of entities and their attributes allowed efficient OSMR operations on large volumes of data. The basic data structure behind relational data model is the concept of a Table. All information in relational databases is stored in tables in which each row represents an entity and columns represent their attributes.

A very important concept commonly used in databases is that there are three distinct levels, shown in Figure 2, at which data must be viewed and managed:

1. **Physical:** At this level the system takes care of storing all data and takes care of accessing, adding, deleing, and updating any particular records that are rows in a table, without other levels having to worry where it is really stored in a storage device.
2. **Logical:** This level represents how the data is modeled by application designers. Thus at this level the system knows what are the entities and their attributes that will be used by the database. Using techniques like Entity-Relationship models, the logical model is designed and then translated to tabular representations in the database. At this level the system is only aware of the models of objects (entities) and their characteristics (attributes) that are used in the system. This level is designed based on the desired functionality that a particular application must have.
3. **View:** A database is used by many different types of users, each of which must have different rights and privileges to access different type of data. Thus, in a university database, an instructor may have access to look at the grades of a every student in his class and modify those, but a student can only see his/her grade and will not be able to modify those. Not all functionality of a database is exposed and made available to every user. Different users see only a subset of data and have different rights with respect to addition, deletion, and updating of the data. Each user thus has just *a view* of the database.

An operation to access information from a database is commonly called a query. A query in a relational database is articulated using a sentence in SQL (Structured Query Language).

Information Retrieval and Search

Information retrieval is primarily a field that addresses techniques for searching for information in documents. In early stages it was predominantly concerned with issues to find information in printed documents. Information retrieval as a field was limited in applications until the emergence of the World Wide Web. With the arrival of the Web, each node on this Web was a document and all these documents were connected on the Web. In a sense one could consider that the Web was a giant document of documents. The nature of the Web is significantly different in many dimensions from traditional documents and techniques for searching information on the Web must be designed considering specific needs of the Web, although this may utilize concepts from classic information retrieval. In this section, we will discuss basic concepts from information retrieval and how some of those are being applied in search approaches.

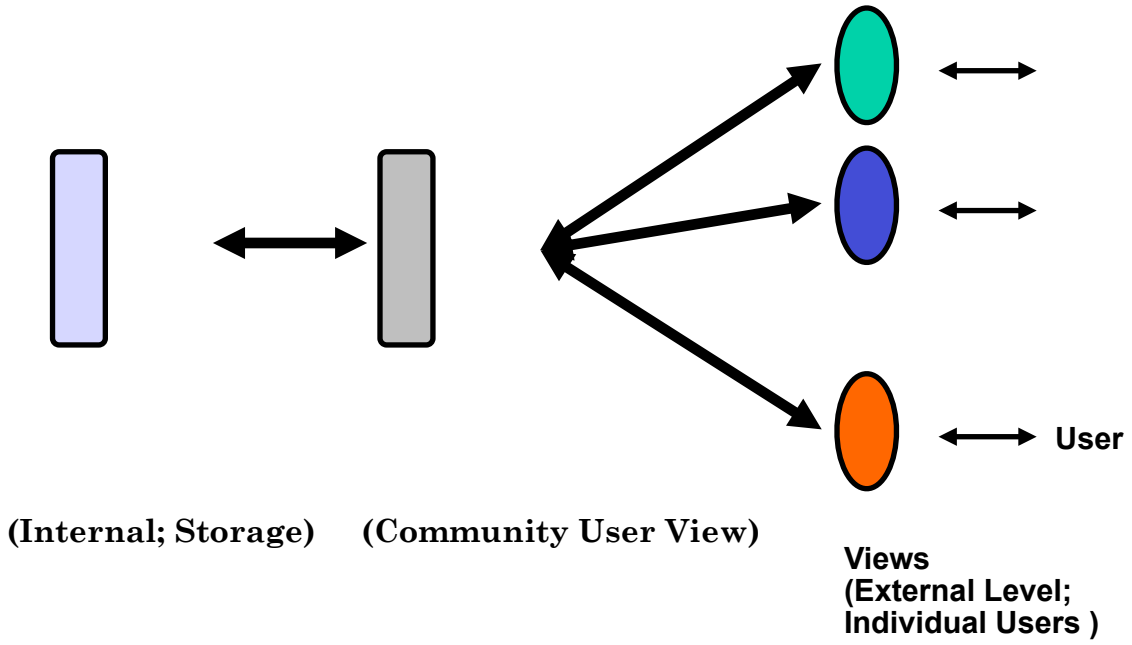


Figure 2: Three Database Levels

In Figure 3, we show the basic architecture of a search approach. This general diagram could be applied to any search problem. Let us discuss information retrieval using some basic blocks from this architecture.

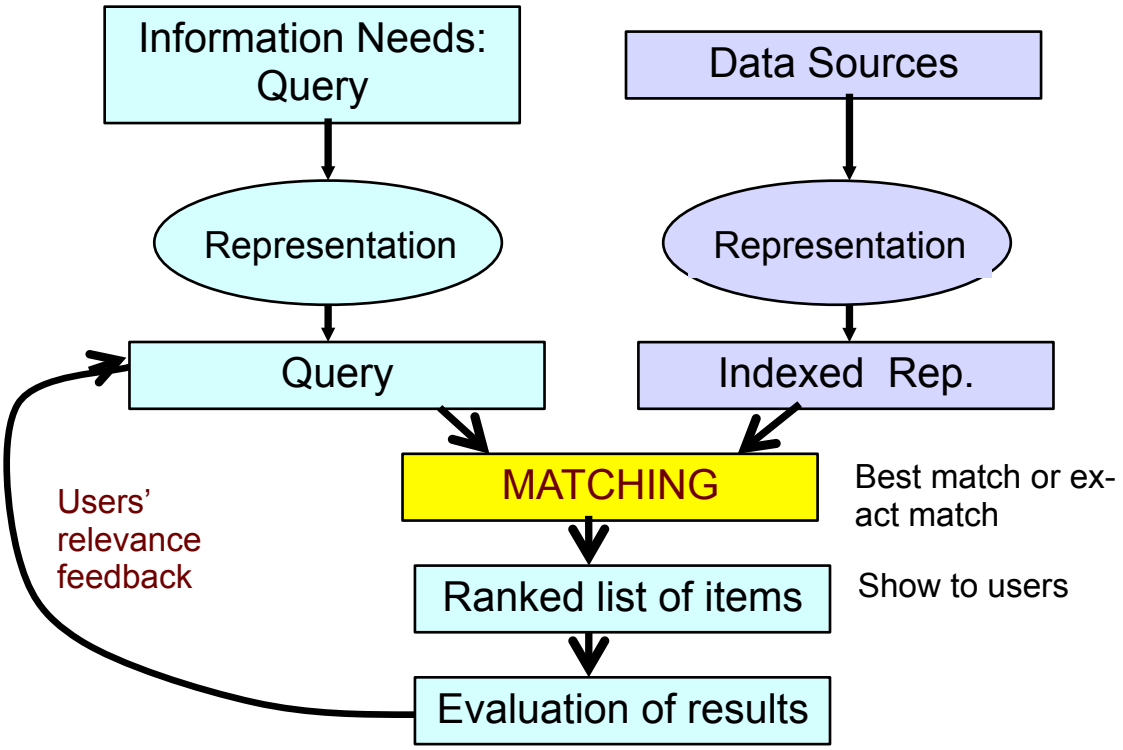


Figure 3: SearchArchitecture

In standard information retrieval data is in the form of a collection of text documents. In the collection there may be many documents. The information retrieval system should identify all documents that will be relevant in the context of a search. A data source in an IR system is a document that is in a file; on a web this could be a web page. Each data source should be represented in a form that will facilitate searching. As discussed above, structuring of data helps in organization and indexing resulting in efficient searches. Since text is not structured data, we should represent each file in a structured data form that could be indexed and searched. *Logical Representation* , commonly called a logical view, is used to represent the document or the original source using the data that captures the essence of the document from the perspective of search. One defines or designs a logical representation based on the types of searches that should be performed by users of the system. A logical representation is designed by considering:

- Attributes or features that can be used by users to define their problem, and
- The system can automatically extract those attributes and features from data sources.

One uses representation to index documents by essentially creating an indexed representation of these features and linking those to original documents. In IR systems, words are used as logical representation because

- Words are understood by people, and
- Words can be indexed using very simple text processing techniques.

A user articulates his information need in terms of a word or some combination of words. All queries must be articulated using the same logical representation as the system uses. In some cases, user's representation may not be exactly at the same level. In those cases, the system should translate user query to the IR system such that the IR system gets the query in same logical representation. The query in logical representation is matched with the indexed representation of the documents. The matching process could be very simple as just finding a simple term or could be very complex as we will see in multimedia cases in some of the following sections. Based on the result of matching, the IR system may find many documents that may satisfy the need of the user to a varying degree. Unlike database systems where search is binary, IR systems do not provide direct records that give users the answer, but they provide sources where the information could be found. Thus, the result of matching only indicates that a particular document may satisfy the user's information need.

The list of documents that satisfy user needs may be presented to the user. In many cases this list maybe long and it may not be a practical idea to present the complete list. In such cases, the list should be ranked based on the relevance of the document to the information need and only a subset of higher ranked documents should be presented to a user. A user may look at the list and may provide feedback to the system in terms of which documents are relevant to the information need and which are not. This information provided by the user, commonly called *relevance feedback*, may be used by the system to modify the query and reused to get new list from the system.

For evaluating the performance of a search system two commonly used measures, discussed in more details in a later section, are recall and precision. Recall characterizes how effective is the system in finding all relevant answers from among those who could be considered relevant in the whole document space. Precision is the measure of accuracy of results among all those that are presented as relevant answers. Precision gives us an idea of how effectively system distinguishes between correct and wrong answers.

Documents and Index

A document is not directly used in the matching operation. An IR system computes logical representation for each document and organizes an index of the logical representation of the documents. This index is used in matching. This is very important step because documents could be semi-structured or even unstructured but the system can structure the index and use it efficiently in search operation.

The inverted file index has become the most popular method to index documents based on their representation as keywords. An inverted file index is basically a list of words and all documents where this word appears. For example, an index at the end of a book is a form of an inverted list.

Ranking Results

Since a query may result in many documents and the results are presented in a list, it is essential to decide in which order the documents should be listed in the results. Most people only see items at the top of the list. Ranking algorithms assign ranking to the result; the list contains documents in decreasing value of their ranks. The rank of a document is judged based on many factors such as

- Number of times the word appears in the document
- Position and fonts used for the word (in header, boldface, size)
- The popularity of the document as reflected by the link structure of the Web.

The rank is assigned using a weighted combination of factors

PageRank: Determining importance of a Document Node

A very important component in ranking results is the importance or popularity of the document (or an image or video as the case may be). To determine the importance or popularity of a page or image or video one may derive inspiration from a commonly used idea that is best manifested in academic circles in the forms of citations. One may consider citations and links in web pages similar in that they both point to a source that is considered relevant and important. A document, is considered more popular depending on the number of people referring to the document, and importance of people referring to the document.

If one could develop an approach that considers all links in all documents, in the case of the Web all pages, and develop an approach that could consider this *link graph* to assign a numerical value of importance to each page, then we could consider the numeric values representing the

relative importance of each page. Let us understand this idea using a simple example shown in figure pagerank.

The Web has its link structure reflected by connections of each node to other nodes using links. Consider a node C. The importance of this node is determined by the incoming links to this node and the source nodes of those links. In this case the weights of the incoming nodes are 40 and 23 and are added to assign the node C the PageRank of 63. Since the node C points to 3 different nodes (D, E, and F), each link gets the weight of 21 by equally dividing the pagerank among all outgoing links from it.

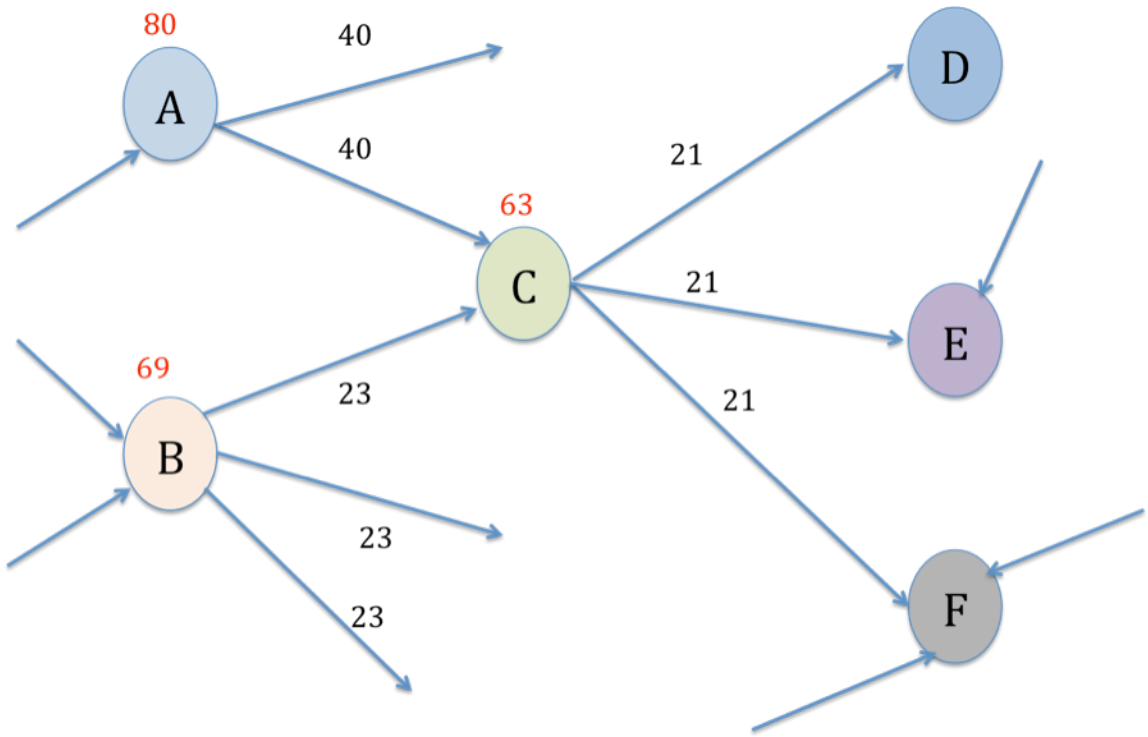


Figure 4: The pageRank of the node C is determined by the weights of the incoming edges from nodes A and B and is distributed equally among the nodes it points to.

This process of assigning pagerank to each node by summing up weights of all incoming links and distributing this among all outgoing is applied to all nodes on the Web, By creating an adjacency matrix to represent the link structure of the Web, and applying eigenvector analysis, computational approaches have been developed to compute the pagerank of each node on the Web.

From Information Retrieval to Multimedia Information Retrieval

MIR has several important components. In Figure MIR, we show a general architecture of a MIR system. We can understand various issues related to MIR systems by considering the role and functionality of each major component as well as interactions among these components. A major challenge for building successful MIR systems is to understand not only each component but its role and interactions in the system. The main components of a MIR system are:

Media processing to extract features: Sensors collect data, but all the processing is done based on information derived from this data, In some cases, the information is at the level of the application, but in most cases, one must rely on intermediate information, commonly called features, as discussed in Chapters X. In every type of signal understanding, feature selection and feature detection is one of the most important steps. .

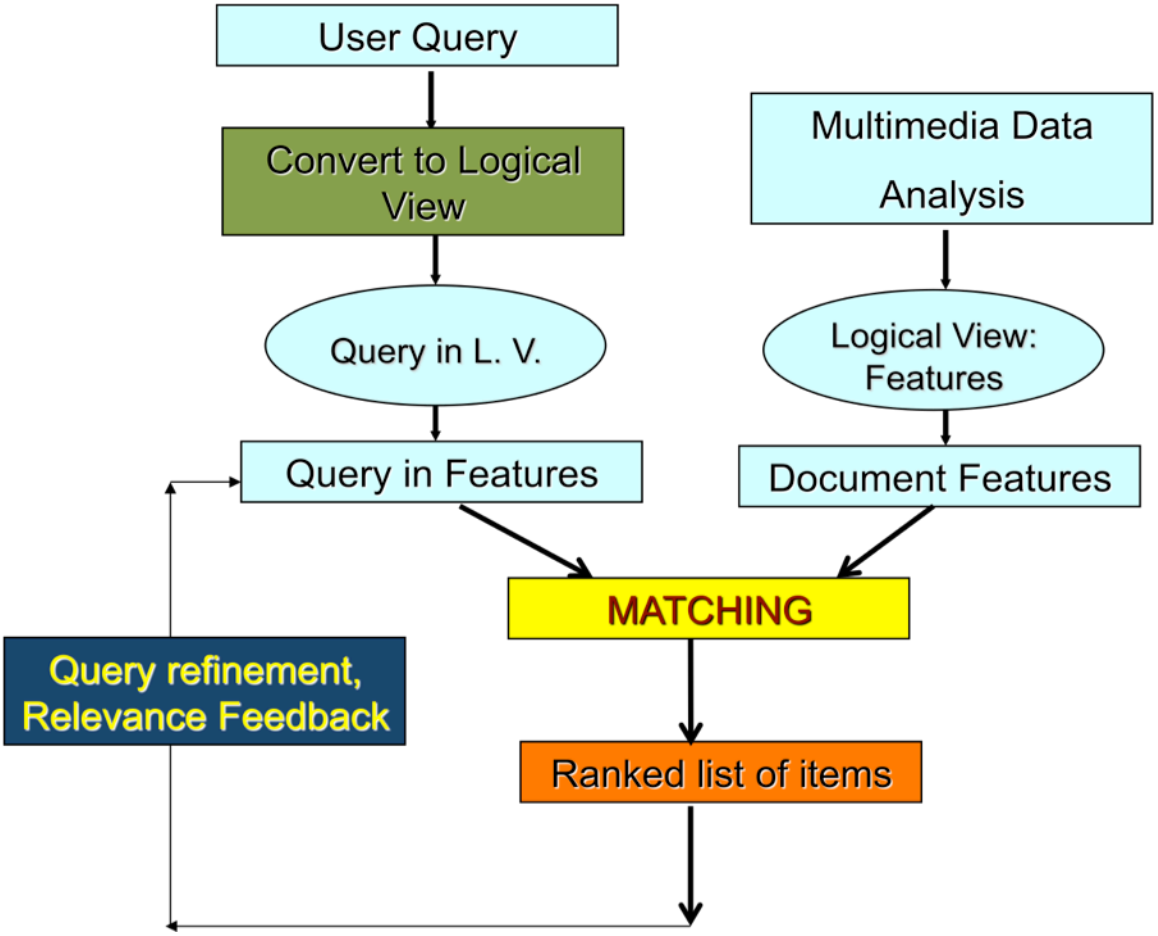


Figure 5: Multimedia Information Retrieval: High-level components.

Storage architecture of Media and Features: Different types of media data have different structure and volumes. The media data should be stored along with connections to features that are extracted from it and application knowledge and information that is derived from it. This multilevel linked structure should be stored to facilitate efficient and effective operations on the data and derived information. An additional factor is that the source of the data and locations where the features and application information is stored may be geographically separated. In fact, increasingly all these locations will be virtualized leading to architecture that will make everything unified for the users at different level.

Indexing: Organization of information means finding suitable approaches to indexing to efficiently gain access to it. In multimedia data, this becomes a serious challenge. In most applications, one is less interested in the medium, than in the information that is derived (usually as a combination of more than one mediums and sources). Current approaches to indexing are strongly influenced by the medium. Unified indexing approaches should allow accessing information of multiple medium based on information requirement. Another problem is the dimensionality of data. Number of features required to characterize an image and other media data is very large and requires multidimensional indexing techniques.

Interaction environment: Commonly a user formulates a query to the system and the system responds by providing an answer. The system is stateless – it does not remember your earlier queries and its answers are independent of the situation of the user. Increasingly systems are trying to introduce some knowledge about the state of the user and are also trying to personalize the responses to the user by using profiles and other information related to the user, as shown in Fig. MIR. Relevance feedback has been used as a mechanism to refine the response of the system for a specific query. Thus, though most systems still utilize a simple stateless query and answer system, many applications are starting to embed search or retrieval systems in their environment to make the whole environment more interactive and contextual. Many applications are naturally designed to use an incremental approach to solve a problem. In such systems, retrieval is a component that serves in the background by bringing in right information to the user at right instant.

Presentation and distribution of multimedia information: Multimedia data is not as natural to ranked list based presentation as text is, thus other techniques are being explored. Increasing use of wireless mobile devices is making this a further challenging problem. With live data in MIR, this problem may become even more challenging.

In the following sections we will try to address these important issues and direction in various MIR systems.

Visual Information Retrieval

A visual information retrieval (VIR) system goes beyond text-based descriptors to elicit, store, and retrieve "imagery-based" information content in visual media. The basic premise behind VIR systems is that images and videos are information-bearing entities and that users should be able to query their content as easily as they query textual documents, without necessarily using manual annotation. VIR combines the analysis component of computer vision with the query component of databases and information retrieval systems To understand VIR issues and techniques, we should address three basic questions:

- What constitutes the "information content" of an image or video in the specific context of any application? Or, what is visual information?
- How can a user specify a search for a desired piece of information?
- How efficient and accurate is the retrieval process?

Two kinds of information are associated with a visual object (image or video): information about the object, called its metadata or context, and information contained within the object, called content and represented by visual features. Metadata is alphanumeric and generally expressible as a schema of a relational or object-oriented database or using XML. Visual features are derived through computational processes—typically image processing, computer vision, and computational geometric routines— executed on the visual object. The simplest visual features that can be computed are based on pixel values of raw data, and several early image database systems used pixels as the basis of their data models. These systems can answer such queries as:

• Find all images in which at least 500 pixels are in the color range represented by (red = 240 to 255, green = 130 to 200, and blue= 0 to 30).
• Find all images that have about the same color in the top half region of the image as this particular one.
• Find all images that are rotated versions of this particular image.

These queries are based on image content but are not related to any objects or concepts. If the user's requirements are satisfied with these, data modeling for visual information is almost trivially simple. However, a pixel-based model suffers from several drawbacks. First, it is very sensitive to noise, and therefore a couple of noise pixels may be sufficient to cause it to discard a candidate image for the first two queries. Second, translation and rotation invariance are often

desirable properties for images. Third, apart from noise, variations in illumination and other imaging conditions affect pixel values drastically, leading to incorrect query results.

These limitations are not to say that pixel-oriented models are not used in visual information retrieval. Significant image and video segmentation requires considering and analyzing pixel attributes. However, multimedia information retrieval based only on pixel values is not very effective because humans usually ask queries in terms of objects and events, rather than pixel attributes.

Content Based Image Retrieval or Image Similarity

Image retrieval techniques were initially developed to consider image collections. These techniques were called 'content based' retrieval (CBIR) because they considered image characteristics for retrieving images. An image could be represented using its basic features. Commonly used basic features are color, texture, and structure or shape.

We discuss these features in more detail in chapter X. These three attributes of an image are captured using different types of image characteristics computed using image processing techniques applied to the pixel characteristics.

The very early versions of CBIR used a weighted combination of the above features to judge image similarity and rank pictures. Sometimes these systems were also called Query by Image Example because the query for an image was another image.

Figure 6 and Figure 7 show two examples of these systems. In both these figures, screenshots for a query and its results are shown. In both figures, the query image is the first image, the left-top image and all images are shown in the order of their similarity to the query image from the database. The number of images in the database for these examples was 20,000.

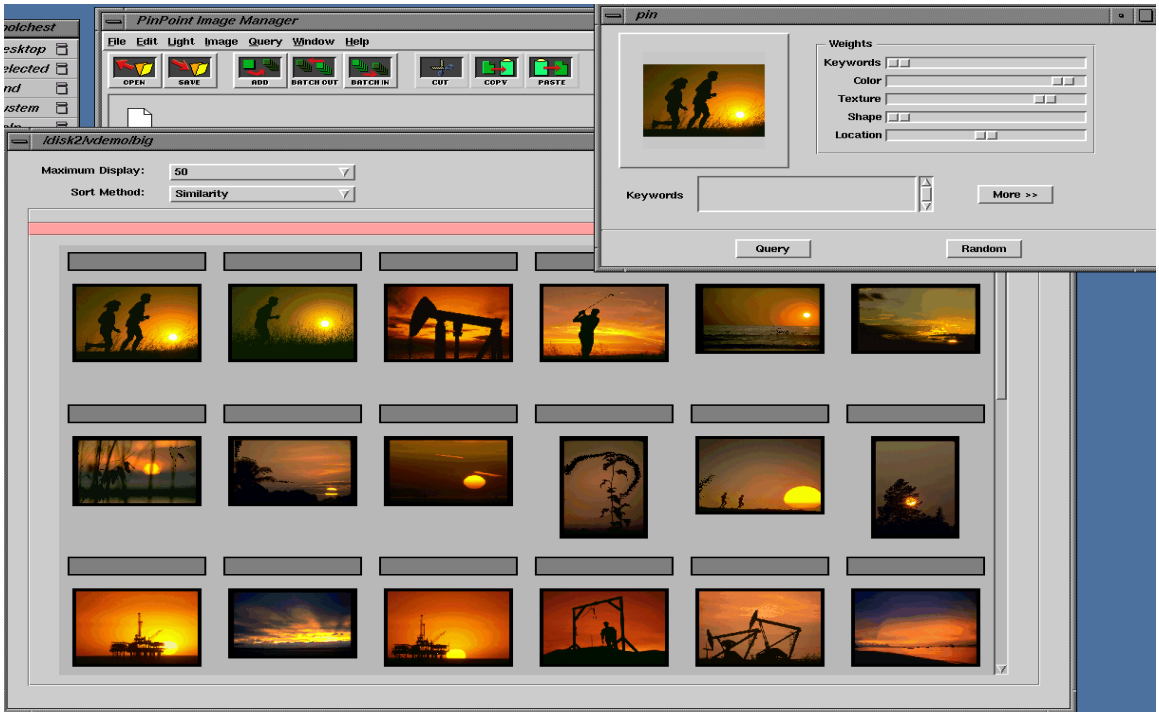


Figure 6: A screenshot showing the selected image, the weights for different features and results corresponding to this query.

Figure 7: For the query related to a rose, in the top left corner as the first result, the top 50 results are shown here.

In Figure 6, at the right top the panel shows that a user could assign importance to different image features in the query.

These systems are ranking images based on similarity of visual characteristics of images; they do not have any concept of objects as we know them. Thus in Figure 7 one can see that most images are similar to the query image because they contain large number of green pixels with similar textures and in the centre is another object of different color – in most cases similar to the color of the query image. The system is not trying to detect roses as in the query image, though majority of results contain flowers, particularly roses.

Many enhancements have been made to these basic techniques. One can divide each image into multiple regions, say a 3 X 3 grid and computer these features for each region and compare these with the corresponding ones. The similarity is then judge based on how close each region is to corresponding region. Another simple modification is to consider color histogram prepared using only coherent colors. This is done by ignoring those pixels whose color value is different from its neighbors. This means that most noisy pixels and pixels on edges are ignored and only those pixels in homogeneous color regions are considered.

Searching Based on the Semantic Content

If the content of images in terms of objects in it and relationships among them is somehow available, then it is possible to develop powerful retrieval techniques that could be called content-based retrieval. Earlier we discussed computing average attributes of pixels in a query image and using those to retrieve similar images based on these attributes, In more realistic situations, we might consider following queries:

- Show all photos of Jay in which he is participating in a sporting activity.
- Show me all photos of Jay with Tarah in Cabo.
- Was Tarah with Jay in Paris?

These queries require recognition of objects and their relationships in photos. These queries cannot be answered without clear object models.

We might also consider following scenarios in which a camera, say a mobile phone camera, is on and the following query is issued:

- What kind of insect is this?
- Is this plant healthy?
- Who is this person?

These queries also require recognizing objects in the field of view of the camera but are more like query-by-example.

Recognition and Annotation

Object recognition has been a very important research area in computer vision for about 50 years. The more general research area of pattern recognition deals with identifying patterns that may correspond to objects, concepts, or activities in data. The fundamental problem addressed in pattern recognition is: Given N patterns (P1, P2, …,Pn) that are models of corresponding objects or concepts, identify which of these patterns are present in a given data set.

Clearly recognition and retrieval are not the same problem, but recognition may play a very important role in content-based retrieval. The queries listed above all require recognition of objects and concepts and relationships among them. This important fact has resulted in application of several recognition techniques to detect objects and annotate images with these objects for retrieval purposes.

The first step in developing recognition approaches is to have strong models for objects. Considering the variability of objects, their appearances, and changes in appearances in images due to different viewpoints, illumination conditions, and climatic conditions, it is very difficult to create models for objects that could be used for recognition in images. The popularity of machine learning techniques in computer vision and multimedia is primarily for simplifying the process of creation of these models that could be used in recognition.

Despite significant efforts in development of automatic recognition techniques, progress in this area has been quite slow. Since the need for organization and retrieval of images has become a real hurdle in the growth of many applications, manual and semiautomatic approaches received significantly popular.

A very popular concept to emerge in many applications has been that of 'tags' for images and video. This concept has been used in many other applications, including in text documents. A person could assign tags to an image to describe it. The tags could be objects, concepts, or names of objects or places. In fact it could be anything that could describe the image and help in management and retrieval of the image. Due to increasing use of images in many applications, particularly in image sharing, use of these tags appeared quite attractive.

Some common problems with manual assignment of tags that must be considered in any tag based retrieval system are:

- Tags are very subjective. A picture is usually assigned different tags by different people.
- Tags are time dependent. Depending on when tags are assigned to a picture the tags could be very different.
- Availability of tags for pictures is random. For most photos people do not assign tags. It is commonly observed that less than 2% photos on most photo sharing sites have tags. Moreover, people do not assign tags to most photos in their own collections.

Video Retrieval

Video is more than just a sequence of images; it has synchronized audio also. Depending on an application, the information content in a video may be more in audio, particularly speech, more in images or equal in both audio and images. In any case, video must be considered very differently from images.

Information content in a video depends on the type of video. Most videos could be considered in one of the following classes:

- Produced Video: Videos such as TV and Movies are produced by professionals following well defined conventions. These videos can be compared to well authored text where there is a well defined structure in document starting from a book to chapters to sub-chapters to paragraphs and to sentences. Produced videos also have such well defined structure.
- Semi-produced videos: Videos such as sports and seminar lectures are semi-produced. They also follow some general conventions, but these are not so rigid.

Amateur Video (Unstructured videos): These are the commonly produced video by most people. Using a video camera people may collect many video segments and may edit them using one of the commonly available video editing systems. Most of these videos do not follow any videography rules.

Video Segmentation

Video segmentation is used in two very different senses. In computer vision and some related fields, video segmentation is used to mean detecting meaningful objects in video. Thus, a car or a person in a sequence must be segmented based on its appearance and motion despite the fact that this object may look very different in different frames of the video. Many techniques, including motion detection, tracking, and structure from motion among others are used for this purpose. This has been a very active research area now for several decades.

In multimedia information retrieval, the above techniques play important role, but the term video segmentation is used to represent structure of video, rather than content and activities in the scene being captured by the video. This structure is defined following concepts developed in video production community. This structure is shown in Fig VideoSemenatation.

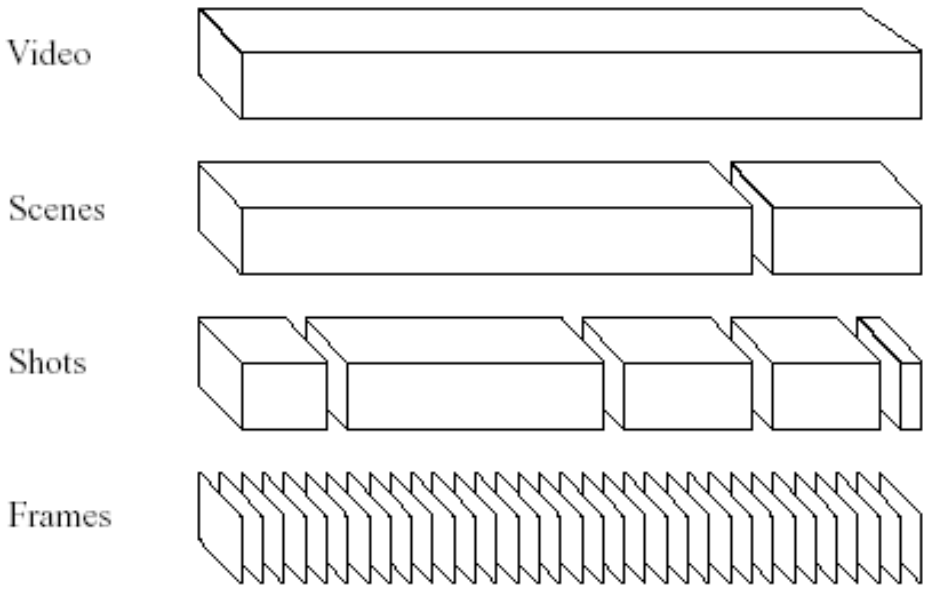


Figure 1: Standard video structuring model

Figure 8: Video could be segmented starting with each frame as the basic unit and then grouping into shots, scenes, and episodes.

We consider video segmentation in the sense of parsing a video rather than partitioning a scene in its individual components. The atomic unit in a video is an image frame. A video contains these frames acquired at regular intervals. The camera is either at a stationary position or in uniform motion to acquire a shot. The shot is considered as a unit of interest in video for building stories. There are many types of shots [Ref] that are used by videographers. In video segmentation, the goal is to group all frames that may form a shot. Many approaches have been developed for automatic shot detection in video. Such techniques are discussed in Chapter X.

A shot in a video is usually represented using one or more 'key frames'. A key frame is simply a representative frame for the shot. For shots of long duration more than one key frame could be

used. The next step in the segmentation is called scene detection. A scene in a video is considered a group of shots that could be considered related to an event or an episode. Thus if two people are discussing then even though the shots may represent alternating between two people, until the conversation is going on, the shots (and the frames belonging to those shots) could be considered to form a scene.

Video Retrieval

Suppose that we have a collection of video. The types of queries that people may ask may be:

- Show me all 3-pointers by Kobe Bryant.
- Show me 3-pointers by Kobe Bryant in the last 5 minutes of the game.
- How did BP finally stop the gushing oil?
- In how many parties, was Tarah wearing Pink dress?
- Show me all romantic scenes from Titanic.
- Show me all stunts by Tom Cruise.
- Show me the car stunts from Casino Royale? What car was Bonds driving?
- List all popular romantic movies in the last decade.
- Show me all players who 'pull' like Tendulkar.
- Show me all stories on BBC that talked about the zsychic Octopus.

All these queries require that the video is analyzed and some information is extracted from video. The queries may be based on

- Objects
- Activities
- Time of activity
- Similarity of activity to a given activity
- Genre of activity
- Visual attributes of objects
- Concepts

And this may require analysis of visual characteristics, audio, and meta-data related to the video.

Integrated Multimedia Information Retrieval

Indexing Single Media or Multimedia using events

Multimedia data has disparate nature with respect to what it represents in terms of physical attributes over space and time. It is captured using different devices and to make sense, it must be

rendered using different devices. The fundamental difference in the nature and volume of data results in the storage of data using different file systems. The capture and rendering technology for each type of media also results in different type of organization of data to match human perceptual perspective of that mode. This results in different approaches to indexing of the data. For example, the video maybe indexed based on the time code, while photos maybe indexed based on photo number and text as page and line numbers. These indexes are fundamentally different and are incompatible. This creates silos among different types of multimedia data. This situation is shown in Figure 9.

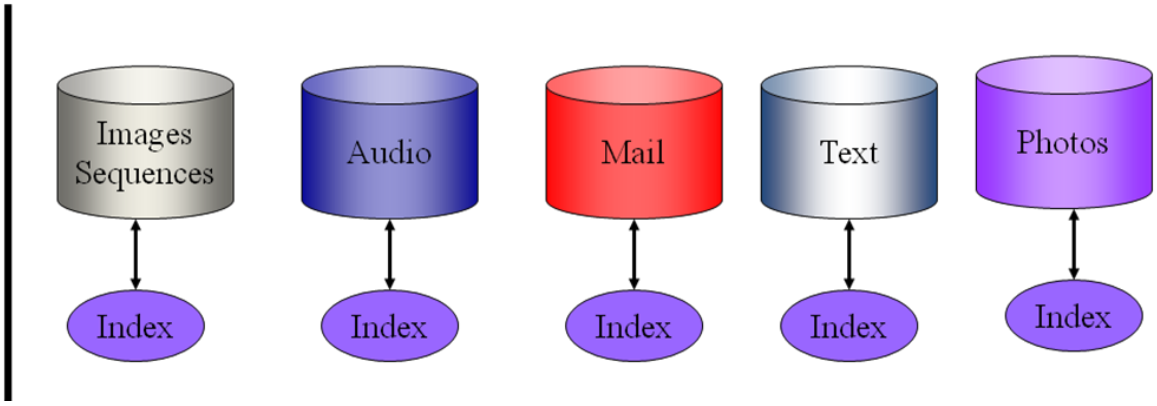


Figure 9: Different type of data and their indexes create isolated silos of multimedia data.

A challenging question for the MIR research is how to break these silos to unify multimedia data?

The only unifying basis behind this data is not in the data but is in the real world. This data is collected for something in the real world that is considered important. Objects and events are complementary in modeling the real world around us. Multimedia, due to its temporal nature, is particularly important in dealing with events. In fact, different modalities of data capture different aspects of events and objects. Since different streams of multimedia data have nothing in common except that they are all related to the same real world event, it is natural to index them using events and their parameters. This is an effective way to unify and index the multimedia data resulting in the situation shown in figure 10. By defining an event model to represent all essential aspects of an event as well as all its associated multimedia, it is possible to create a unified approach to indexing multimedia data.

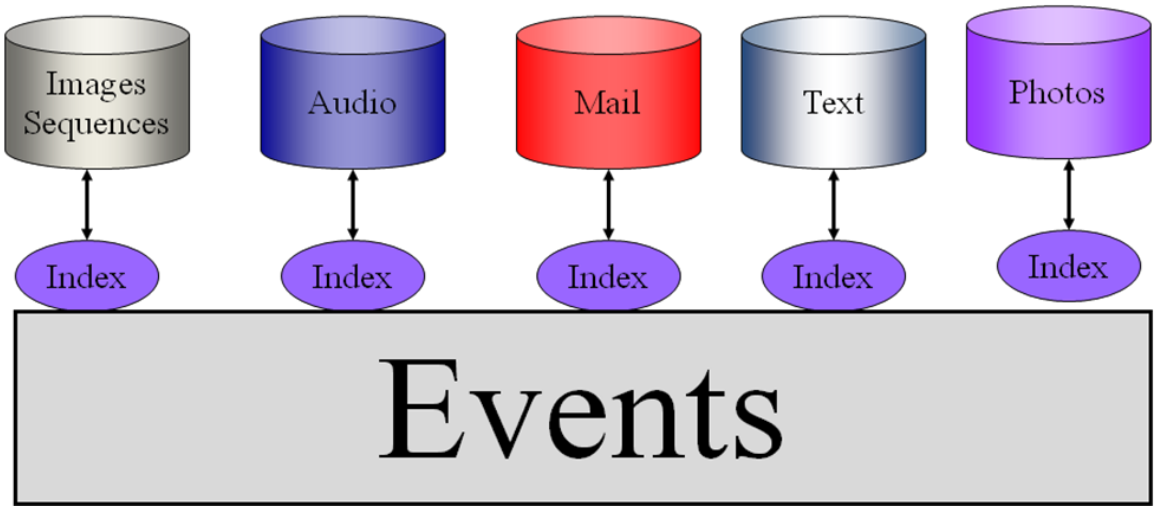


Figure 10: Real world events could be used to unify different media and provide a unified framework for indexing multimedia data.

Relevance Feedback in MIR

In many cases, articulation of a query precisely to get answers from an information retrieval system is not easy. This is because the system may have only a vague idea about what a user is looking for, or the concept or information sought may be difficult to precisely articulate. For example, a user may be looking for people who look like Abraham Lincoln. How can one articulate a query to an image retrieval system to find all people who look like Abraham Lincoln? To deal with such situations, the concept of successive refinement of queries later popularized as relevance feedback was introduced. As shown in Figure 11, based on the query by a user, the system produces ranked list of results. The user then can provide explicit or implicit feedback to the system by conveying which of the results listed are not relevant to the query posed by the user. The system can then modify the query and fetch a new set of results.

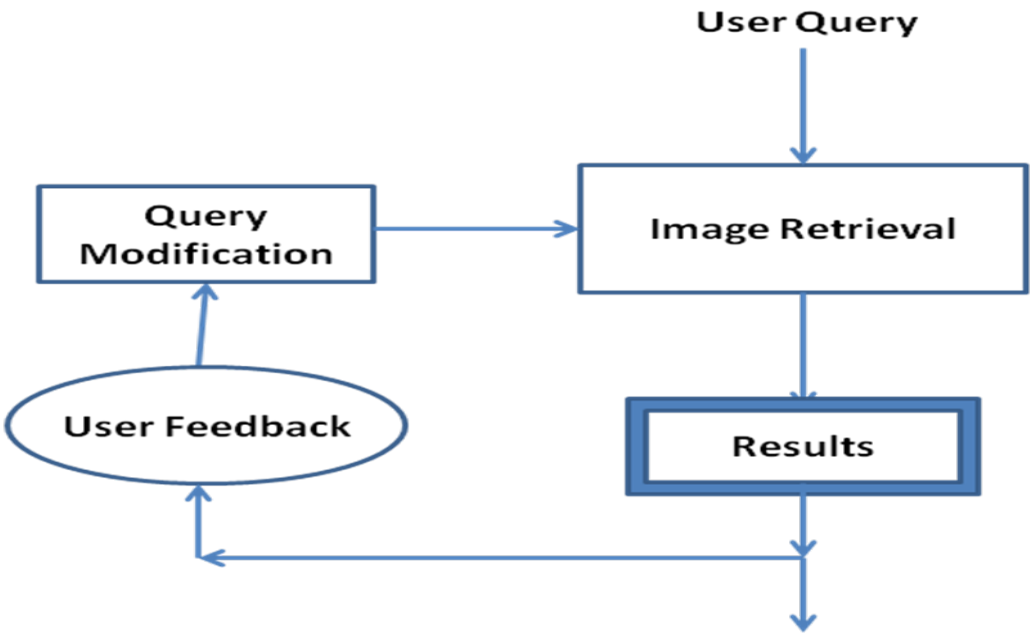


Figure 11: A user query may not represent what the user really has in mind. An iterative query refinement may be used to refine the results to suit user needs.

A simple way of looking at this approach is to consider this as a simple classification problem, as shown in Figure 12. Based on the query by a user, the system considers all documents, or images, in two classes: relevant documents and not-relevant documents. The classification is done using a mathematical approach that considers different features. If the user is not satisfied with the results, then the classification function used by the user and by the system are considered different. The goal of the relevance feedback is to get more information from the user about the classification function that the user has in mind. Based on the feedback from the user, the system changes its classification function to align it better with that of the user.

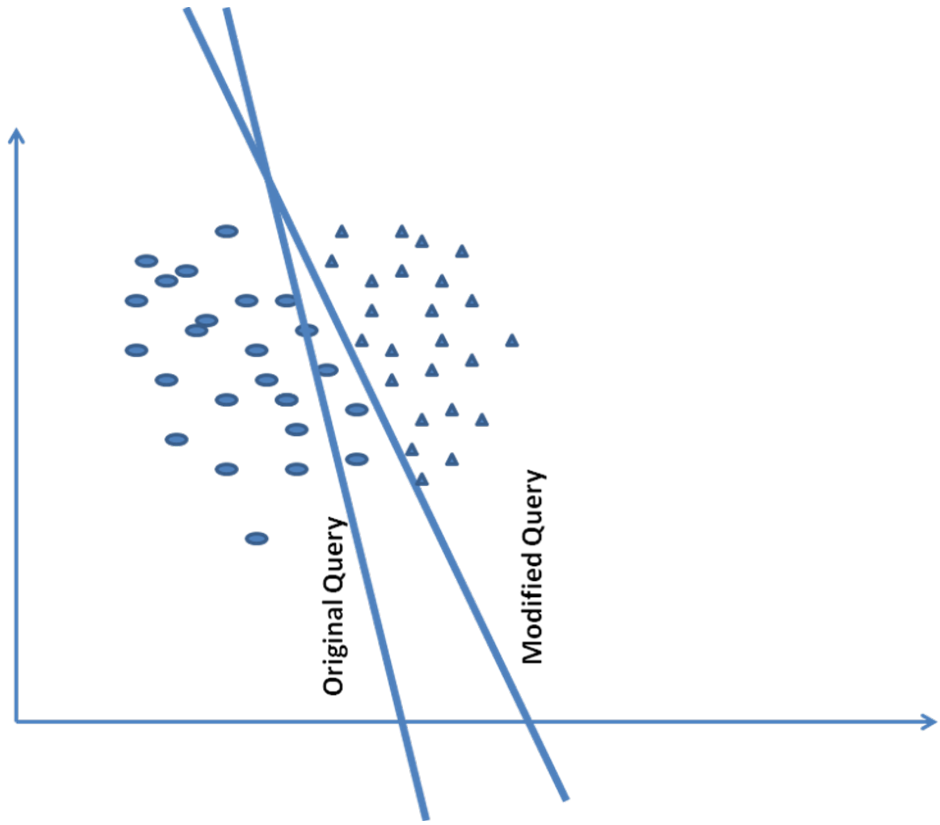


Figure 12: Based on the user query, the system uses a classification function that gives unsatisfactory results. User's feedback about relevance results in a modified query that gives better results.

The above discussion using classification function works very similarly if in place of classification function, we consider a ranking function.

Summaries and Storytelling

The number of photos, videos, audios, and text descriptions of events has increased very rapidly such that the sheer volume of even personal collection of photos has become tedious to manage and maintain. In this section we discuss two related but different problems. Though we will be discussing using more examples from photos, the discussion will be equally applicable to other media and to the combination of these.

Summarization

As in text documents, a summary of multimedia content represents a condensed version of the information contained therein. Consider personal photos. Suppose that you go on a trip to Brazil and come back with 3000 photos that you took. You want to share those photos with your

friends and family. Would you show them all 3000 photos? In 2015 due to the easy availability of digital camera, storage, and ease in transferring those photos to your collection, you take 25000 photos. At the end of the year, you want to send a 'Year in Photos' to your mother, your significant other, your best friend from childhood, your professional friends, and your cousins. How do you select 25 photos to represent your year in photos? These are just two simple cases of summarization of photos.

In simple terms, photo summarization can de described as: Given a set containing N photos, how can one select M (where M is much less than N) photos that represent the set the best. One may consider summarization of photos as a semantic sampling problem such that the goal is to sample the photos, by selecting particular photos, that allow representation of semantics of the complete set satisfactorily. Unlike the sampling problem in signal processing there are two differences here. First, the photos themselves represent sampling of the event by the photographer, and it is difficult to capture semantics quantitatively to evaluate whether sampling is adequate or not.

Given the importance of the problem, many approaches are being proposed for summarization of photos.

Storytelling

One of the most popular and frequent art in human society has been storytelling. From oral traditions to sophisticated video production and now multimedia environments, people have used all possible technology to enhance communication of their experiences. With advances in multimedia, storytelling has taken new directions. We discussed several tools that one may use in story telling in the chapter on multimedia authoring tools. In this section we discuss techniques that will assist people in storytelling using the media that they collect. A storytelling approach could be described as the one shown in Figure 13.

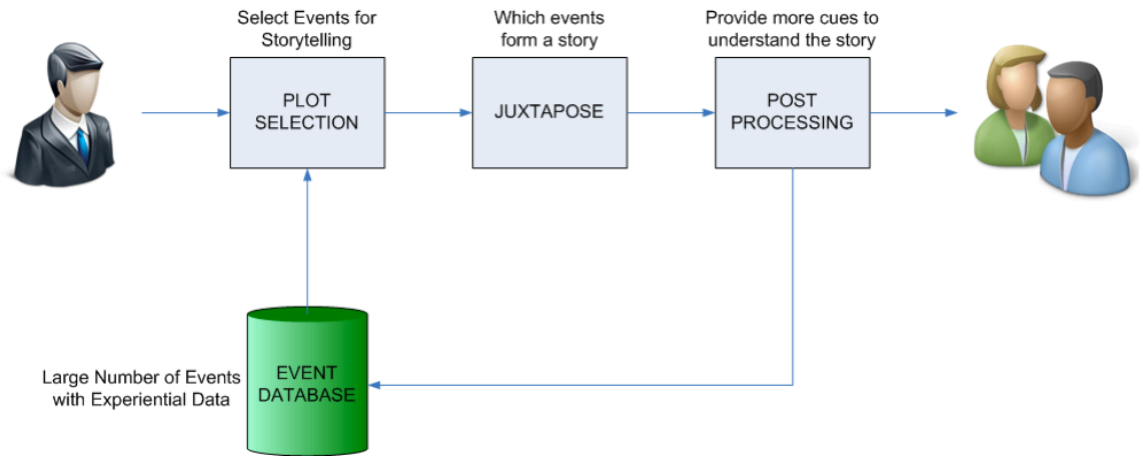


Figure 13: In a storytelling system, an event or a sequence of events is presented to audience using all data that is available. The main steps in such a system, manual or automatic, are shown in this figure.

Storytelling is based on a database of events and the experiences associated with events. A storyteller usually has a message or a perspective that she wants to convey using appropriate events and experiences associated with those. Another important factor in storytelling is the listener or the audience of the story. Events and experiences are selected based on the audience interest and profile also. Essentially, one may define storytelling as given a set of events, and associated media, select appropriate events and for each event select appropriate media data that will maximize the message communication from the storyteller to the audience. This becomes a two stage process. In the first stage one selects appropriate events based on the message and then constructs a coherent story considering the profile and the level of audience.

Further Readings

Multimedia information retrieval is currently (in 2010) one of the most active research areas in multimedia. The growth of content in multimedia requires that tools to organize and index the content for rapid and relevant access must be developed. Starting in early 1990s, this field has grown substantially. Earliest concepts in this area were introduced in [5]. Research in query by pictorial example [21] started a trend that later became known as query by content. [Despite a vibrant research community and very active pursuit, the problems have been challenging. The most fundamental issue in organizing and accessing content is the semantic gap. Theemantic gap was first discussed in [9] and later popularized in [10]. Many research papers are now addressing semantic gap. Many review papers have been written in this field. Interested readers are advised

to start with one of these to get a good feel of the field. A particularly influential paper seems to be [10] that summarized research until year 2000 and is one of the most cited papers. Some later review papers [18, 19, 20] give a good summary of MIR research and are good complements to the earlier review [10].

Query by successive refinement was introduced in [3] and later was formalized in general cases using the concepts of relevance feedback in [17]. The concept of emergent semantics was introduced in [4] and is likely to attract more attention as the data size is increasing exponentially. Some trends to organize personal photos using context are explored in [2,6,7,13]. A good summary of metadata associated with photos is in [11] and how to use those is explored in [1]. Some researchers have started exploring concepts of visual words [12]. One must be very careful in using words in multimedia because words in text are manually delimited and are used as defined in a dictionary. Both these are not currently valid for visual words.

Use of events for unified access to multimedia was presented in [23] using models of events defined in [22].

Index Terms

Exercises

References

1. Sinha, P., Jain, R.: Semantics In Digital Photos: A Contenxtual Analysis. In: Proceedings of the 2008 IEEE International Conference on Semantic Computing, IEEE Computer Society (2008) 58–65
2. Anguera, X.; Xu, J.; and Oliver, N.: Multimodal photo annotation and retrieval on mobile phones, Proceeding of the 1st ACM international conference on Multimedia information retrieval, 2009.
3. J. Bach, S. Paul, and R. Jain, "An Interactive Image Management System for Face Information Retrieval," *IEEE Transactions on Knowledge and Data Engineering, Special Section on Multimedia Information* Systems. Publication. 1993.
4. Simone Santini, Amarnath Gupta, and Ramesh Jain "Emergent semantics through interaction in Image Databases" IEEE Transactions on Knowledge and Data Engineering, summer 2001.
5. A. Gupta, T. Weymouth, and R. Jain, "Semantic Queries with Pictures, The VIMSYS Model," *Proceedings of VLDB'91, 17th International Conference* on *Very Large Data Bases*, Barcelona, Spain. Sept. 3-6, 1991

6. Bo Gong and Ramesh Jain, “Segmenting Photo Streams in Events Based on Optical Metadata”, Proc. IEEE Conf. on Semantic Computing, Irvine, CA, Sept. 17-19, 2007.
7. Ling Liu and Tamer M. Özsu (Eds.) (2009). "Encyclopedia of Database Systems, 4100 p. 60 illus. ISBN 978-0-387-49616-0.
8. Sinha, P., Jain, R.: Classification and annotation of digital photos using optical context data. In: Proceedings of the 2008 international conference on Content-based image and video retrieval, ACM (2008) 309–318
9. Santini, S. and Jain, R. “Beyond Query by Example; IEEE Second Workshop on Multimedia Signal Processing, Redondo Beach, CA , **pp.** 3 – 8, USA 7-9 Dec 1998.
10. **Content-Based Image Retrieval at the End of the Early Years**
11. Found in: IEEE Transactions on Pattern Analysis and Machine Intelligence
12. Arnold Smeulders , et. al., December 2000.
13. EXIF: Exchangeable image file format for digital cameras: Exif version 2.2. Technical report, Japan Electronics and Information Technology Industries Association, 2002.
14. K. Barnard and D. Forsyth. Learning the semantics of words and pictures. In *Proc of IEEE Intl Conf Computer Vision*, volume 2, pages 408–415, 2000.
15. M. Boutell and J. Luo. Bayesian fusion of camera metadata cues in semantic scene classification. In *Proc. IEEE CVPR*, 2004.
16. M. Boutell and J. Luo. Photo classification by integrating image content and camera metadata. In *Proceedings of ICPR*, 2004.
17. M. L. KHERFI AND D. ZIOU AND M. BERNARDI **Image Retrieval From the World Wide Web: Issues, Techniques, and Systems**
18. GUPTA, A. AND JAIN, R. 1997. Visual information retrieval. Commun. ACM 40, 5, 70–79.
19. RUI, Y., HUANG, T. S., ORTEGA, M., AND MEHROTRA, S. 1998. Relevance feedback: A power tool in interactive content-based image retrieval. IEEE Trans. Circ. Syst. Video Technol. 8, 5, 644–655.
20. M. S. Kankanhalli and Y. Rui, “Application Potential of Multimedia Information Retrieval”, Proc. IEE, April 2008.
21. R Datta, D Joshi, J Li, and J. Wang, “Image Retrieval: Ideas, Influences, and Trends of the New Age”, ACM Computing Surveys, VOl 40, No. 2, April 2008.
22. M. L Kherfi, D. Ziou, and A. Bernardi, “Image Retrieval from the World Wide Web”, ACM Computing Surveys, VOl 36, No. 1, March 2004
23. M Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yonker, “Query by Image and Video Content: The QBIC System”, IEEE Computer, Sept. 1995.
24. G. Utz Westermann and Ramesh Jain,” Towards a Common Event Model for Multimedia Applications”, in IEEE Multimedia, January 2007.
25. Ramesh Jain, “Out of the Box Data Engineering: Events in Heterogeneous Data”, Key note talk in Proceedings of International Conference on Data Engineering, Bangalore, India, March 2003.

Chapter 18: Statistical Methods for Multimedia Content Analysis

Multimedia content analysis tries to infer from the content of multimedia, i.e. from images, audio, or video with the goal of enable interesting applications. For example, finding all the smiling faces in a video to extract portrait pictures of them is an application of multimedia content analysis. Another example is search based on keywords spotted in the audiotrack of a movie. Later chapters will explore some of the applications of multimedia content analysis, such as information retrieval further. In this chapter we will provide a short introduction to the basics of machine learning that are vital to understand research work in multimedia content analysis.

Historically, even though the mathematical foundations are very similar, until recently multimedia content analysis research seemed to be strictly divided according to the type of data that were to be analyzed. Therefore many researchers work on either acoustic processing, computer vision, or natural language processing. Only recently multimodal content analysis has emerged trying to merge the different types of sensory data to create unified *multi*media approaches that can benefit from the synergy of leveraging modalities in a combined way.

The foundations of audio, speech, image, and video content analysis are rooted both in the signal processing community, which is part of electrical engineering, as well as the field of statistics, which is part of mathematics. Many terms that are still used have been inherited by these two fields. A newer field, which has come up with the raise of the computer, is called machine learning. It is a subfield of statistics, paired with some biology and neuroscience roots. The field of machine learning redefines many terms originally created in statistics and biology. The application of machine learning together with the foundations of signal processing to different kinds of data created the different fields of audio, speech, image, and video processing, which by themselves created new terms. There are different reasons for those fields to have become separated, some are social and very pragmatic. A very important one though is the amount of data that has to be processed. Audio and speech processing is the oldest field because computers where already able to process speech in the 60s and 70s. Image analysis is a slightly newer field, and video analysis is the newest because there is much more data to be processed. With different maturities of the fields, different generations of people have worked on the different types of data and hence, different vocabulary is used.

Today, the processing capabilities of modern computers allows us to think about approaches that analyze multimedia multimodaly, i.e. processing audio, images, motion, and meta-data synergis-

tically. A combined processing promises improved robustness in many situations and is closer to what humans are doing: The human brain takes into account not only patterns of illumination on the retina or periods of excitation in the cochlea, it also combines different sensory information and benefits from past experience. Humans are able to use context information and to fill in missing data by associating parts of objects with already learned ones.

The following chapter provides a short introduction to the field of multimedia content analysis before the next chapters dig deeper into the specialties of individual sensory modalities.

Basics of Machine Learning

The basic workflow of a machine learning approach is illustrated in Figure 1.

Using training data, relevant features are extracted and passed to a machine learning algorithm. These algorithms, are usually methods derived from mathematical statistics and produce statistical models. These are basically sparse representations of the data that allow thresholding of any kind. To train the models, the right answers have to be provided which are given in the ground truth data. This is usually metadata created by human annotators. Currently Common algorithms include Gaussian Mixture Models, Neural Networks, Support-Vector Machines,

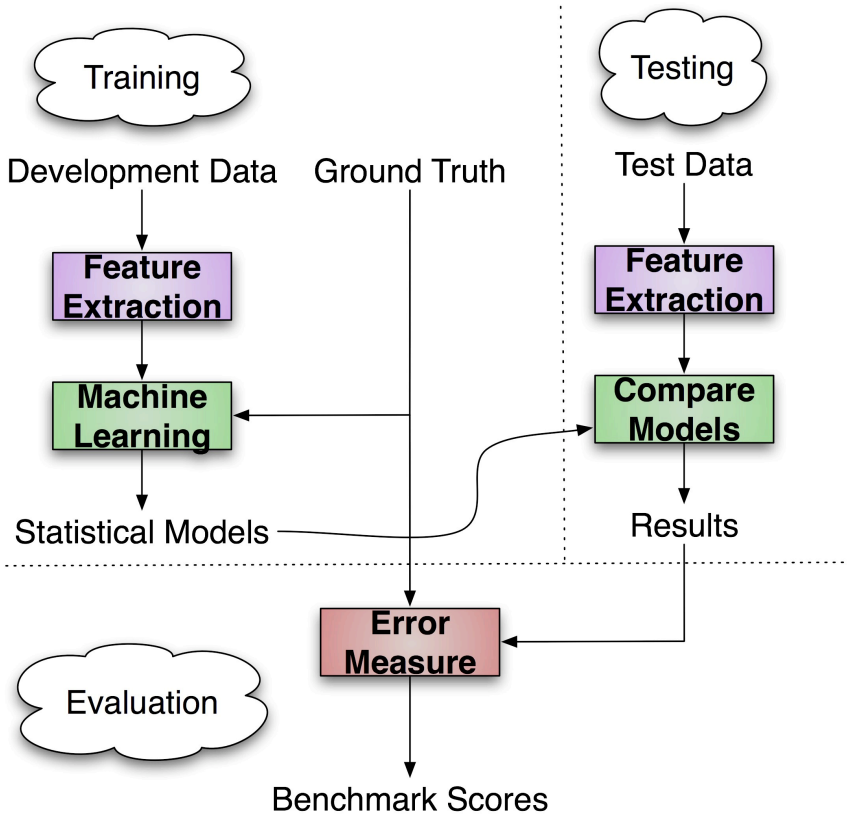


Figure 1. Typical workflow of content analysis

Decision trees (for example ID3), k-Nearest Neighbors, Bayesian Networks, and Hidden Markov Models. We will describe the basic ideas of these algorithms below. Feature extraction methods are usually derived from signal processing or electrical engineering, we will describe common features later.

In testing mode, the statistical models are then used to perform the actual content analysis task. The test data is run through the same feature extraction process. The results are either just used for the actual application or compared against the ground truth to benchmark the quality of the algorithm.

Here is an example: Let's assume we want to detect the gender of a speaker by their voice. The development data consists of samples of female and male speech from humans of different ages uttering different text. From the raw audio files, features are extracted that try to reduce the amount of raw data while emphasizing speaker characteristics (features are further explained in Chapter XXX). These are then used to train a statistical model for the two classes. In testing mode, a random audio snippet containing speech and undergoing the same feature extraction is then presented to the model for classification. The output may be used for anything. In evaluation mode, however, the quality of the classification is measured by comparing the classification results with the ground truth for many test cases and counting both the number of right and wrong classifications. In our example, an error might be expressed as percentage of the wrongly classified test samples versus the total number of test samples.

Unsupervised machine learning, as opposed to supervised learning, omits the training step and statistical models are created on the fly using the test data. Evaluation is performed in a similar way.

Supervized Modelling

In the following, we will discuss some common supervized learning techniques often used in multimedia. Supervized learning uses a development set to learn a model for the mapping of sample values to classes (classification task) or a model for sample values to a continuous range (regression task). A typical classification task is the answer to the question "does this portion of audio contain speech?" or "which digit can be seen in the picture?", a typical regression task is "what are the geo-coordinates of the position of the camera in this image?". The list of presented algorithms here and the explanations are not exhaustive. As explained above, machine learning is it's own field and interested you are invited to explore the literature for further reading.

k-Nearest Neighbors

The most basic learning technique, which can almost not be called learning technique is nearest neighbor search. Let's assume one has a dataset with a number of samples in an N-dimensional space e.g., 8x8 binarized scanned digits (compare also first example in Chapter XXX). Each sample is labelled, i.e. we know to which class it belongs. A nearest neighbor search assumes that the dataset spans an N-dimensional space. This space is our ``trained" knowledge from the development data.

Now, given a new sample without label, i.e. from the test data, we search the development data for samples with low distances from the sample. We then take the labels of the k samples with lowest distances and vote: The label that is in the majority wins. Obviously, with *k=1* the decision is always clear and also it is usually be a good idea to choose an odd number for *k* even though, depending on the number of classes, this will not always guarantee an inambigous decision. A rule of thumb is that *k=1*, *k=3* or *k=5* are good numbers. If these often result in two many different candidate labels to make a decision, either the distance metric or the k-nearest neighbor algorithm is the wrong choice. Nearest neighbor search with *k=1* was also referred to as the post-office problem, referring to an application of assigning a residence to the nearest post office. See literature.

The following pseudo code describes a k-nearest neighbor search with an arbitrary distance metric.

```
// Input: A number N of labelled samples S with label set L,
  // k number of neighbors to take into account,
// f distance function,
// x, a sample to be classified
// Output: A label for x
kNN(S, L, N, k, f, x)
    D := [] // array of distances with length = k
    I := [] // array of indizes to neighboring samples with length k
    C := [] // array of counts for a label
    FORALL d IN D  // initialize all distances to infinity
          d := +infinity
    FORALL c IN C // initialize counts to zero
          c:=0
    FOR i:=0 to N
          distance := f(x,X[i])
          FOR j:=0 TO k-1
                IF (distance<D[j])
                      D[j]:=distance
                      I[j]:= i
    FOR i:=0 TO k-1
          C[L[I[i]]]++ // increase count for label
    Y:=max_index(C) // Take the label with the maximum count,
                    // note: decision might be ambiguous
```

```
kNN := Y
```

Obviously, the quality of the classification results will depend on many factors, including the chosen distance metric which determines when two samples are considered similar. Also, it can been shown that under a broad set of conditions, as dimensionality increases, the distance to the nearest data point approaches the distance to the farthest data point. Therefore, higher dimensional data usually requires different techniques. More importantly, the algorithm does not create a generalized model for the classes, i.e. a small cluster of outliers can decrease the quality of the classification result. Also, the kNN algorithm requires comparing the test sample with each sample in the training set, which can be quite expensive. These thoughts lead us to the next algorithm.

k-Means and k-Gaussians

We already presented the k-Means algorithm in Chapter XXX (compression) as a vector quantization algorithm. By keeping the labels intact, k-Means can be a good solution to reducing the size of the development set for kNN as comparing to the means instead of the whole data will reduce the number of comparisons. Also, the means can be seen as abstract representation of the data in that they represent more than one individual value and therefore make the comparison less prone to classification errors due to outliers. k can be chosen to match the number of classes in the development set or increased for finer granularity.

When using the means to model data, it is often desired to have an indication of the variance of the represented cluster as well i.e. how close to the mean is the typical sample in the cluster. Also, as can be observed in nature, often cluster values have the typical bell-curve shape. Therefore, a very frequent way of modeling data beyond k-Means is modeling it with k Gaussians. A function f for a Gaussian is defined as

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

with μ being the mean and σ^2 the variance. Figure 2 shows the typical bell-shaped curves for different means and variances.

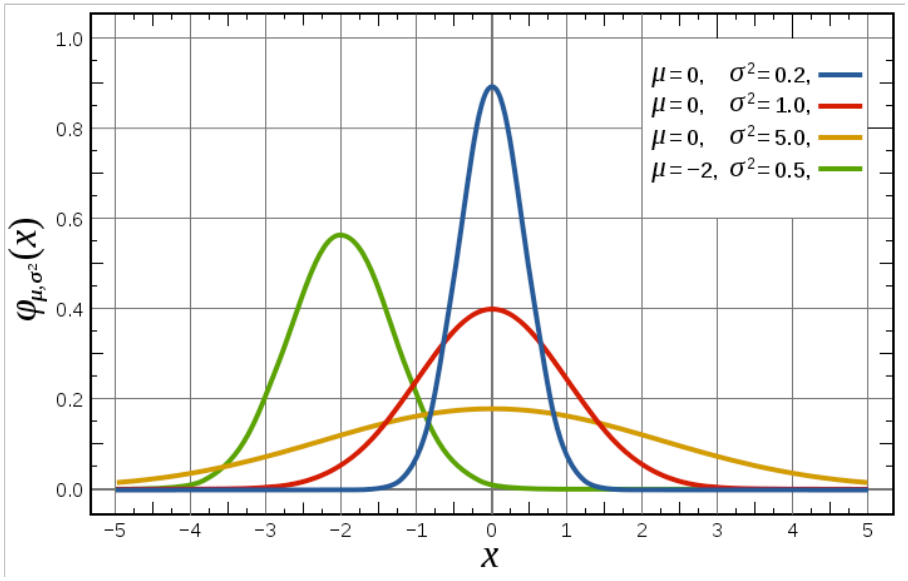


Figure 2. A set of Gaussians with different μ and σ^2.

Modeling data with k Gaussians only requires a slight modification of the k-Means algorithm: After the means have been determined, the variance σ^2 has to be calculated. This can be done by interpreting the sample values as observations as defined by probability theory. The Gaussian function becomes a probability density function:

$$f(x;\,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-(x-\mu)^2/(2\sigma^2)}\ ,\ x \in \mathbb{R}.$$

and the set of samples are interpreted as values of a discrete random variable X. The variance

$$\sigma^2 = \sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2$$

with probability mass function $x_1 \mapsto p_1, \ldots, x_n \mapsto p_n$,

In order to classify a test sample, the probability of the sample belonging to each of the k Gaussians is calculated and the Gaussian with the highest probability is chosen. The pseudo code for calculating k-Gaussians in analogy to k-Means is left as an exercise.

Mixture Models

More than often, clusters do not exactly adhere the Gaussian function. For example when there are two or more "bumps", i.e. the distribution is multimodal. This is especially the case when clusters are the mixture of two or more independent components. For example, a color model for face images might consist of the mixture of a model for the skin, a model for hair, a model for

lips, and one for eyes. Obviously, different components will have different importance values, which can be expressed by weighting the individual components. In general, a probability mixture model f_X is defined as a weighted sum of its component distributions:

$$f_X(x) = \sum_{i=1}^{n} a_i f_{Y_i}(x)$$

with X being a discrete random variable and a mixture of the n-component discrete random variables Y_i for some mixture proportions $0 \leq a_i \leq 1$ where $a_1 + \cdots + a_n = 1$.

Like k-Gaussians and k-Means, a Gaussian Mixture Model can also be learned by Expectation Maximization (EM). As explained earlier, EM is an iterative method for finding most likely parameters for models (when probabilistic models are used, the likelihood is maximized). The algorithms alternates between the so-called expectation step (E-Step) which estimates the expected likelihood given the current model, and a maximization step (M-Step), which computes new parameters given the current memberships, maximizing the expected log-likelihood found on the E step. These new parameter-estimates are then used to determine the distribution of the memberships in the next E step. This is repeated until the system converges based on an evaluation metric or a fixed amount of iterations. The mixing coefficients a_i are the means of the membership values over the N data points:

$$a_i = \frac{1}{N} \sum_{j=1}^{N} y_{i,j}$$

The model parameters f_{Y_i} are calculated by using the data points x_j that have been weighted using the membership values. For example, for a Gaussian, μ is calculated using:

$$\mu_i = \frac{\sum_j y_{i,j} x_j}{\sum_j y_{i,j}}.$$

The implementation of Gaussian Mixture Model training is left as an exercise. The concept of Gaussian Mixture Models is rather straightforward, at the same time, Gaussian Mixture Models are a very powerful concept used in many places in multimedia research, especially in visual and acoustic object recognition (e.g., in interactive image segmentation and speaker identification). In practical approaches, the number of Gaussians per cluster, the number of clusters (unsupervized case), and the number of training iterations and/or the convergence criterium are important factors that have to be empirically determined. Using too few Gaussians may make the like-

lihood estimation poor and therefore not work well for discriminating between the different clusters on the test data. Using an unlimited number of clusters or Gaussians in a mixture model allows to model an arbitrary distribution. However, using too many Gaussians results in so-called overfitting of the data, for example when each data point is represented by its own Gaussian, the models are too specialized and not useful for representing the unknown test data. In other words, there is a trade-off between the number of parameters that should be used to describe the model given the number of training samples and the complexity of the distribution: To few parameters result in a bad model, too many parameters result in a bad abstraction. Chapter XXX already describes the Bayesian Information Criterion, which is one way of objectively approaching the trade-off.

Artificial Neural Networks

An Arificial Neural Network (ANN), under computer scientists often simply called "Neural Network" (NN), is another statistical modeling technique. The name stems from its motivational background of trying to simulate some of the structure and/or functional aspects of biological neural networks. ANNs usually consist of an interconnected group of components (often called artificial neurons) of which each of them simulates one function. The artificial neurons, i.e. the functions, are connected in layers with each group of neuron feeding its output to a corresponding neuron in the next layer. In the end, the goal is to model a function f:X->Y, with X being the input data and Y typically being a desired regression or classification output. In most cases, an ANN is an adaptive system that learns its structure from the training data by assigning weights to the connections and parameters of the neuron functions. The most common type of ANN used in practice is the so-called Multi-Layer Perceptron (MLP). A Multi-Layer Perception is an ANN composed of many interconnected Perceptrons.

A Perceptron is a basically treshold function which maps its input x (a real-valued vector) to an output value $f(x)$ (a single binary value) across the matrix.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

where w is a vector of real-valued weights and is the dot product (which computes a weighted sum), b is the so-called bias, a constant term that does not depend on any input value. So the value of $f(x)$ (0 or 1) is used to classify x as either a positive or a negative instance. If b is negative, then the weighted combination of inputs must produce a positive value greater than $|b|$ in

order to push the classifier neuron over the 0 threshold. Perceptrons take multiple inputs x_i by summing them together:

$$f(x) = f(w_0x_1 + w_1x_1 + w_2x_2 + \ldots + wmxm) + b)$$

As can be seen, a single Perceptron therefore can divide the input space into two areas, for which one area is classified as 1 and the other as 0. The division is a line in two dimension, an area in 3 dimensions, and a hyperplane in in any higher dimensionality. The Perceptron is therefore said to be *linearly separating* the space. The following algorithm can be used to to train the parameters *w* and *b* given a set of input points and a set of classification labels for the input points (training data).

```
// Input: Training data with n samples x[i] and desired output labels l[i]
// Output: Perceptron parameters w[i] and b
p_learn(x[i],l[i], n)
    Initialize the w[i] and b randomly by
    setting w[i][t] (0<=i<=n) to be the weight at iteration t, and
    b to be the bias in the output node, and
    w[0]:= -b, and x[0]:=1, and
    settting w[i][0] to small random values.
    t := 0
    DO
        FOREACH sample x[i]
            // calculate output
            output[t]:=
                (w[0][t]*x[0][t]+w[1][t]*x[1][t]+...+w[n][t]*x[n][t])+b
            w[i][t+1]:= w[i][t]+gamma*(l[i] - output[t])*x[i][t]
            // where 0<=gamma<=1 is a value to slow down the adaption rate
        t := t+1
    UNTIL t>threshold
p_learn := (w[i],b)
```

Practically, two problems have to be addressed for Perceptrons to be applicable to everyday classification problems. First, the Perceptrons have to be able to also separate spaces that are not linearly separable but require more complicated function. Second, the weights and biases should be tuned automatically based on a training set. The following paragraphs will shortly explain how to do both, however the theory of Artificial Neural Networks is an ongoing field of research. and many other networks exist in addition to the basic ones presented here. See literature for more details.

In order to be able to separate input space into more than two classes and also to separate a problem space using more than linear separation several Perceptrons are connected allowing piece-

wise linear approximation of the separation function. A so-called Multilayer Perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate output. It is a modification of the standard linear Perceptron in that it uses three or more layers of nodes (so-called neurons) that are interconnected, i.e. the output of the neurons in one layer feeds the inputs of the neurons in the next layer. The typical structure looks like the one sketched in Figure 3.

The input layer is the layer of nodes receiving the input directly. The output layer is the layer that outputs the functions directly. The so-called hidden layer is the layer in between. One can show that an MLP with three layers can approximate any function given enough neurons in the input, output, and hidden layers. Figure 3 shows an example of an MLP calculating the binary XOR function. For solving multimedia classification problems, many more hidden nodes are needed. A practical rule of thumb is that the number of total parameters in a Neural Network should be about 1 per 10-20 parameters to be learned.

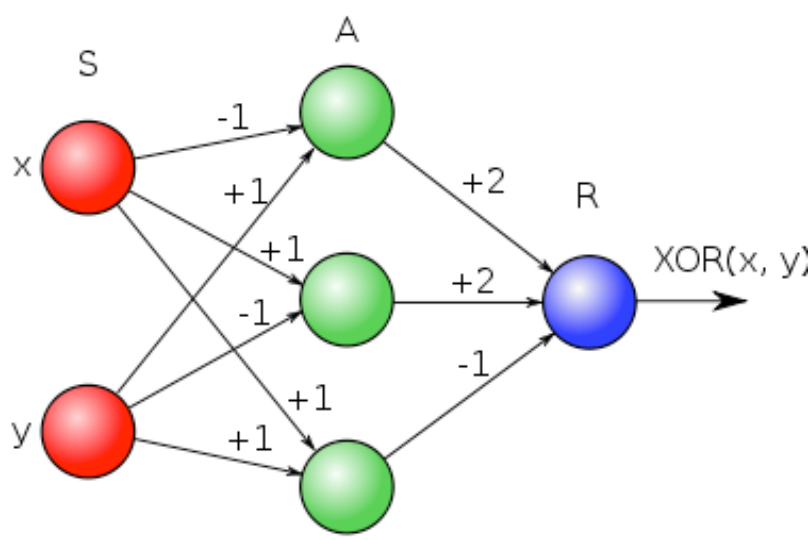


Figure 3. A Multilayer Perceptron (MLP) for calculating the binary function XOR. The numbers on the edges define the weights *w*. The bias *b=0*.

While the network in Figure 3 is still straightforward to design manually, MLPs with hundreds or thousands of hidden layers are not. Therefore, different ways of learning the weights based on training data. The most well-known algorithm is called backpropagation. The idea is that the weights are learned by calculating the output given the current weights and then calculating the difference between the output and the training data. When doing this for many samples, one can extrapolate an error function given the weights and then tune the weights following the gradient of that error function to a minimum. To be able to calculate the gradient both the error function

and the Perceptron function must be differentiable. Therefore, mean-square error is often used to compare output and training data and the Perceptron function is often replaced with the so-called soft-max function, which is defined as:

$$p_i = \frac{\exp(q_i)}{\sum_{j=1}^{n} \exp(q_j)},$$

with p_i being the values of an output node, $q_{i,j}$ are the net inputs to the output nodes, and n is the number of output nodes. It ensures all of the output values p are between 0 and 1, and that their sum is 1. This also makes it easier to interpret the outputs of a node as probabilities.

The following constitutes pseudo-code for the back propagation algorithm:

```
// Input: An MLP network with hidden and input weights wh and wi,
// Training data T
// Output: A network with new weights
backprop(network(w[i]),T)
    Initialize the weights in the network randomly
    DO
          FOREACH sample e in T
            output := compute_output(network,e) // forward pass
            mse := mean-square error (T[e] - output) at the output layer
            gradient := mse * e
            FOREACH wh in the hidden layer:
                Compute delta(wh) // backward pass
                wh := wh - gradient
            FOREACH wi in the input layer:
                Compute delta(wi) //backward pass
                wi := wi - gradient
    UNTIL mse<threshold
backprop := network
```

Support Vector Machines

Another approach to solve the supervized classification problem is the so-called Support Vector Machine (SVM). Like Neuronal Networks this is more a class of algorithms rather than a single one and they are constantly evolving. Therefore the reader is, again, strongly encouraged to follow the most recent literature on this field of specialization. Like the Multilayer Perceptron, a SVM performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

Not only is the above equation familiar, SVMs are closely related to neural networks: An SVM using a sigmoid kernel function is equivalent to a two-layer MLP. In contrast to MLPs, SVMs regularly solve only two-class problems. The following paragraphs explain the main concepts.

The goal when using an SVM is to find the optimal separation with a hyperplane that discriminates two clusters of input data in such a way that the sample within one category of the target outcome are on one side of the plane and cases with the other category are on the other size of the hyperplane. The vectors closest to the hyperplane are called the *support vectors* as they define the separation margin: SVM analysis finds the hyperplane that is oriented so that the margin between the support vectors is maximized. Figure 4 illustrates the idea based on a two-dimensional example.

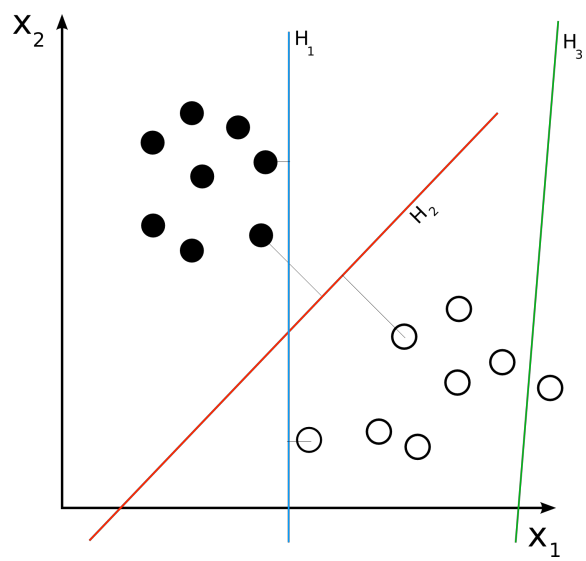


Figure 4. Different (hyper-)planes in space. H_3 does not separate the space correctly. H_2 and H_1 do, but only H_2 maximizes the margin between the two classes (and the support vectors).

A simple two-dimensional case can be solved using dynamic programming or the Perceptron learning algorithm (see above). However, as explained earlier, often data is not linearly separable. While MLPs implicitly learn to fold the input space when training the hidden layer of the network, SVMs use a different trick: They use a so-called kernel function to manipulate the input

values so that they may be separated by a hyperplane. In other words: Rather than seeking for a non-linear separation function, they transform the value space to be linearly separable. A major trick is to increase the dimensionality of the input space. This trick *always* allows to find a linear separation if the dimensionality is chosen high enough. Figure 5 illustrates the idea.

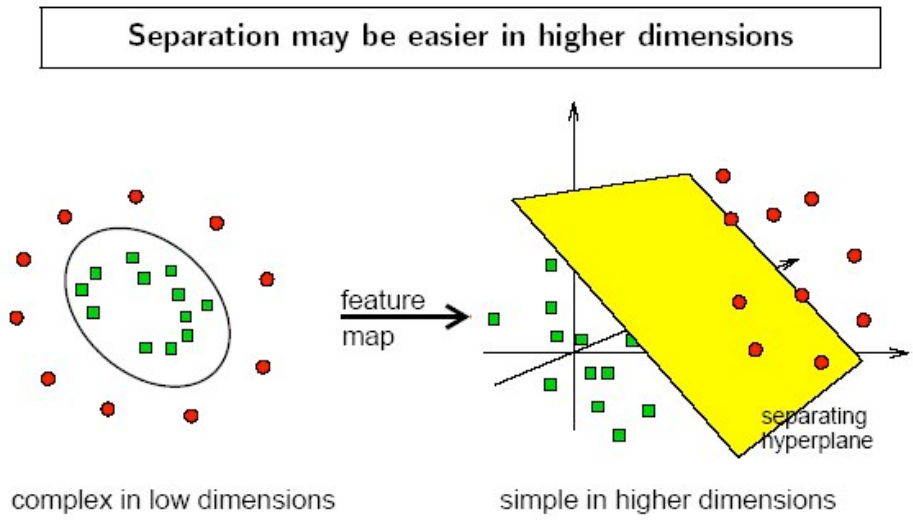


Figure 5. Separation of the input space is always easier in higher dimensions. TODO: REDO IMAGE. (Copyright status not clear)

Mathematically, a kernel can transform any algorithm that solely depends on the dot product between two vectors. Wherever a dot product is used, it is replaced with the kernel function. Thus, a linear algorithm can easily be transformed into a non-linear algorithm. In practice, many kernel mapping functions may be used and finding the right one can be tricky. However, a few kernel functions have been found to work well in for a wide variety of applications and seem to predominant in scientific literature. The most simple of them is the linear kernel:

$$K(x, x_i) = x_i^T x,$$

with x_i being the input values. The two most frequently used one are probably the polynomial kernel

$$K(x, x_i) = \left(1 + x_i^T x / c\right)^d,$$

and the radial basis function (RBF) kernel

$$K(x, x_i) = \exp\left(-\left\|x - x_i\right\|^2 / \sigma^2\right),$$

where d, c, and σ are constants. If one chooses a sigmoid function as kernel, such as

$$K(x, x_i) = \tanh\left(k\, x_i^T x + \theta\right),$$

with k and θ being constants again, one can emulate an MLP (as outlined above).

Finally, to allow for some flexibility in separating the categories, SVM models can have a cost parameter C, that controls the trade off between allowing training errors (e.g. allow some points to be wrongly classified) and forcing rigid margins. This is called *soft margin classification*. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

What we need now is an algorithm to train an SVM. In general, all it involves is solving the following optimization problem. As explained so far, a general SVM can be expressed as

$$u = \sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) - b$$

where u is the output of the SVM, K is the kernel function which measures the similarity of a stored training example x_i to the input x, $y_i \in \{-1,1\}$ (given by the training labels) is the desired output of the classifier, b is a threshold or bias, and a_i are weights which blend the different kernels. Training therefore consists of finding the weights a_i, which is usually expressed as a minimization of the following formula:

$$\min_{\vec{\alpha}} \Psi(\alpha) = \min_{\vec{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(\vec{x}_i, \vec{x}_j) \alpha_i \alpha_j \qquad \sum_{i=1}^{N} \alpha_i,$$

subject to constraining the range of the weights and the constraint that

$$\sum_{i=1}^{N} y_i \alpha_i = 0.$$

The most common algorithm to solve the problem is the so-called Sequential Minimal Optimization (SMO) algorithm, invented by John C. Platt (see references). The main idea behind the algorithm is, again, to break up the large problem of optimizing the margin into a number of smallest possible problems, thereby achieving a O(n log n) runtime. The algorithm is described in pseudo-code in the original reference (XXX).

Markov Chains

The machine learning algorithms described so far are all trying to classify or model particular patterns or samples of data independently of each other. Especially when handling multimedia data, however, it is very often that case that a particular sample depends on the previous sample. For example, when parsing English language in the form of text or speech the probability of a vowel followinga 'y' is much higher than the probability of a 'j' following a 'y'. These kind of temporal dependencies can be modeled in several ways, including with ANNs. However, the most common way in literature is the the use of a Markov chain, and more particular the use of a Hidden Markov Model (HMM).

A Markov chain is a stochastic model that assumes the Markov property, which is that if the conditional probability distribution of future states of the process depend only upon the present state. In other words, given the present, the future does not depend on the past. Generally, this assumption enables reasoning and computation with the model that would otherwise be computationally intractable. A Hidden Markov Model (HMM) is a Markov chain for which the state is only partially observable, i.e., observations are related to the overall state of the system, but they are typically insufficient to precisely determine the state. A typical problem is that one is given a sequence of observations and has to compute most-likely corresponding sequence of states. Figure 6 shows a very common example from literature.

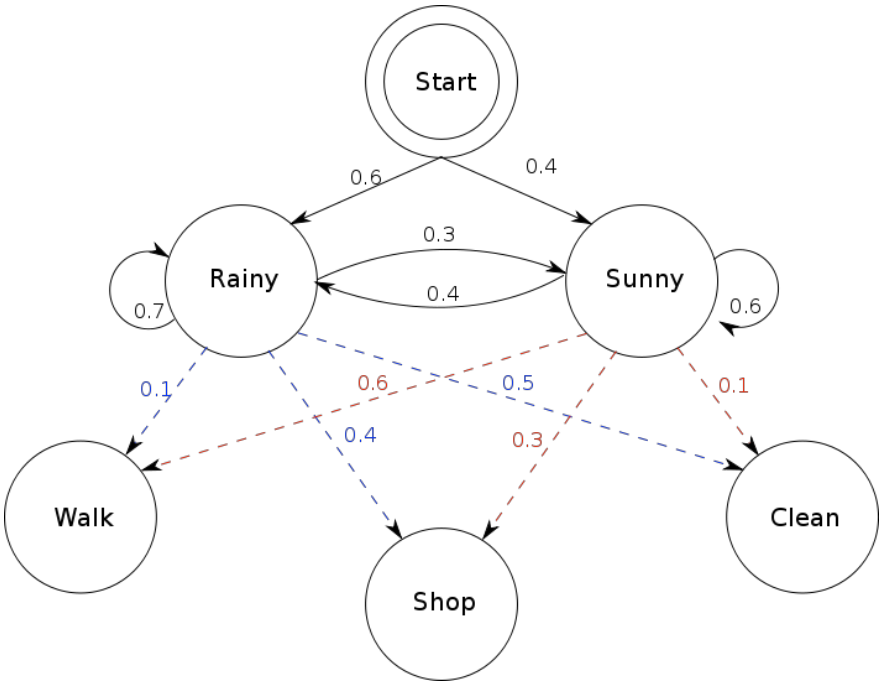


Figure 6. An example HMM modeling activities given weather probabilities.

Assume two people living far apart from each other but talking over the telephone frequently about what they did each day. One of them is only interested in three activities: walking on the beach, shopping in a mall, and cleaning her own apartment. The choice of what to do is determined exclusively by the weather on a given day. The other person on the phone has no definite information about the weather and also doesn't dear to ask but knows general trends. Therefore, based on the activity reported each day, the other person tries to guess what the weather must have been like.

To do this, the weather and activities are modeled in a Markov chain as depicted in Figure 4. There are two states for the weather, "Rainy" and "Sunny", which as pointed out in the description cannot be observed directly. In other words, they are hidden. However, on each day, there is a certain chance that the person on the phone will perform one of the following activities: "walk", "shop", or "clean", which, as we know, depend on the weather states. Since we know about the activities, we call these states observations. Based on the general trend of weather (climate) and the statistics of the average type of activity, one can label the edges of the HMM with probabilities.

In this model, the probabilities emitted by the start node represent the belief about which weather state of the HMM is in based on climate. The probability edges between the weather nodes represent the transition probability of the weather, in other words how likely is the weather to stay the same or change. In our example, there is only a 30% chance that tomorrow will be sunny if today is rainy. Finally, the so-called emission probability represents how likely it is that a certain activity is performed each day given the weather. In this model, there is a 50% chance of the cleaning apartment activity when it is rainy and a 60% chance for the walking state when it is sunny.

In order to understand how a HMM helps with temporal modeling let us consider this example:Out two phone chatter talk to each other three days in a row. The activity sequence looks like this: walk-shopping-cleaning. The question now is: What is the most likely sequence of rainy/sunny days that would explain these observations?

This is very frequent problem, especially when modeling language. An analog for speech recognition would be: Given the observation of the likelihoods for certain phones, what is the probability of the speaker having uttered word x?

Given an HMM, the answer to this problem is given by a very popular algorithm, named after his creator Andrew Viterbi, the so-called Viterbi algorithm. The idea behind the algorithm is that builds a graph of all possible state transitions. Then, the path with the maximum probability is chosen, the so-called Viterbi path. The following pseudo code shows the algorithm:

```
// Input: A sequence of observations obs[], hidden states hstates[],
// start_p is the start probabilty, transp[] and emit_p[] the transition
// and emission probabilities, respectively.
// Output: The output probability of the most likely path.
viterbi(obs, hstates, start_p, trans_p, emit_p)
  graph = []
  path = []

  // initialize
  FOR y IN hstates
      graph[0][y] = start_p[y] * emit_p[y][obs[0]]

  FOR t:=1 TO LENGTH(obs)
      newpath = []
      FOREACH y IN hstates
           prob := max([(V[t-1][y0] * trans_p[y0][y] * emit_p[y][obs[t]])
                   over all y0 in hstates
      graph[t][y] = prob
   prob := max((V[LENGTH(obs) - 1][y]) over all y in hstates
viterbi := prob
```

Decision Trees

A decision tree is a decision support tool that uses a tree-like graph structure to model decisions and their possible consequences, sometimes including the probabilities of event outcomes, resource costs, and other factors. Figure 7 is a photograph taken from Wikipedia showing a whiteboard with a typical decision tree. A decision tree is a widely-used tool in many areas. In multimedia, decision trees are particularly useful to combine different media. For example, different sensors might output different values for different input (e.g. audio classification determines car noise, video classification determines a street -- what is the probability for a highway scene?). In order to combine the sensors, a decision tree might help determine the final outcome based on the different measurements. Often, the trees are created manually. However, learning them automatically based on training data is often desirable because the tree might become rather complex. Again, the topic of decision tree learning is an ongoing field of research and we cannot present it exhaustively in this book. Therefore, we limit ourselves to one of the most used algorithms, called C4.5.

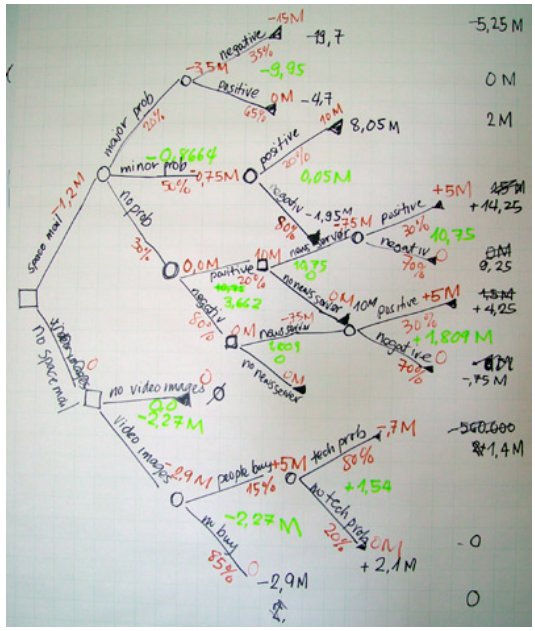


Figure 7. A manually created decision tree on a whiteboard.

C4.5 has been created by Ross Qinlan and can be used for classification. It is used very frequently in many multimedia projects, mostly for analyzing the combination of textual features. The idea of the algorithm is straightforward: The tree is build iteratively and at each node feature ranges are split so that they most effectively separate its set of samples into subsets enriched in one class or the other. In order to find which subset would "enrich" a class more, C4.5 defines the notion of information gain, which in turn is based on Entropy. The information gain over a split is:

$$G(S,A) = E(S) - \sum_{i=1}^{m} f_S(A_i) E(S_{A_i})$$

where $G(S,A)$ is the gain of the subset S after a split over the A features, $E(S)$ is the information entropy of the subset S, m is the number of different values of the features A in S, $f_S(A_i)$ is the frequency of the items possessing A_i as value for A in S, A_i is *ith* possible value of A, S_{Ai} is a subset of S containing all items where the value of A is A_i. $E(S)$ is defined as follows:

$$E(S) = -\sum_{j=1}^{n} f_S(j) \log_2 f_S(j)$$

with n is the number of different values of the features in S (entropy is computed for one chosen feature), $f_S(j)$ is the frequency of the value j in the subset S. The following pseudo code demonstrates the idea:

```
// Input: A sequence of observed examples[], a sequence of attibutes[],
  // a target attribute ta.
// Output: A labelled decision tree T with children labelled +/-
c45(examples,attributes,ta)
    T := root_node
    IF all e in examples > 0:
          root_node.label := "+"
          return T
    IF all e in examples < 0:
          root_node.label := "-"
          return T
    IF number of predicting attributes is empty:
      return the single node tree Root, with label := most common value
      of the target attribute in the examples
    FOREACH attribute a
          Find the information gain from splitting on a
    A := The attribute that has highest information gain
    Decision tree attribute for root = A.
          FOREACH a of A:
                Add a new tree branch below root,
                corresponding to the test A = a.
                Let examples(a), be the subset of examples
                that have the value a for A
                IF examples(a) is empty
                      Then add a leaf node with label :=
                      most common target value in the examples
                ELSE add the subtree C45(Examples(a), attributes - {A}, ta)
c45 := T
```

Since trees can become very complex, the C4.5 algorithm defined a second routine that iterates through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes. Sometimes, features can be associated with a cost function to return different information gains depending on their use. This may be especially interesting when combining different sources of sensor data, since they may have different validity.

Unsupervized Modelling

All the methods explained so far assumed the presence of training data. Sometimes, however, training data is not available or it is not practical to ask users to provide it. Still, it might be desirable to label multimedia data according to their similarity. This is called clustering and examples of clustering were already explained in Chapter XXX (lossy compression): The k-Means and X-Means algorithms are perfectly suitable for these tasks.

Often, however, because of the type of data one is dealing with it is desirable to use the same machine learning algorithm one would use in the supervized case, for example, when certain machine learning methods have proven to work well with a certain type of feature. Fortunately, all

one has to do is to replace the error function, which is based on comparison with ground truth training data in the supervized case, with a similarity function for the unsupervized case. The following pseudo-code illustrates Gaussian Mixture Model training (as discussed earlier) for the unsupervized case:

```
// Input: A sequence of n samples x[], a number of clusters k
// Needs: k-Means, Gaussian function f(x,mu,sigma), sum()
// Output: k parameter triples (a[],mu[],Sigma[]) that describe k
  // weighted Gaussians
GMM_train(x[],k)
  Initialize a[],mu[],Sigma[] by using k-Means clustering
  Initialize probabilities p[][]:=0
  iter:=0;


  DO
    // E-Step
    Compute probabilities p[i][j]:=(a[j]*f(x[i],mu[j],Sigma[j]))/
                                   (sum(a[j]*f(x[i],mu[k],Sigma[k])))
                for all elements in x[i:=0..n] and all clusters j:=0..k.
    // Needs to be stored in an n×k matrix. Each individual cell of this
    // matrix corresponds to probability that xi belongs to the Gaussian
    // distribution specified by mu[k],Sigma[k].

    // M-Step
          Update the weights a[j]:=sum(p[i][j]/n) for j:=0..k
          Update the centroids mu[j]:=sum(x[i],p[i][j])/sum(p[i][j])
          for j:=0..k
          Update the co-variances
          Sigma[j]:=sum(p[i][j]*(x[i]-mu[j])(x[i]-mu[j]))/sum(p[i[[j])
          for j:=0..k
    iter++;
  WHILE (iter<threshold)
GMM_train := (a[],mu[],Sigma[])
```

A typical general pattern for unsupervized training is bottom-up and top-down agglomerative hierarchical clustering. One way to think of the concepts is to think of them as sorting algorithms: A divisive approach can be implemented recursively like quicksort. An agglomerative approach is comparable to bottom-up mergesort. So in general, for divisive segmentation/ clustering we start at the top with all frames in one cluster. The cluster is split using a flat segmentation algorithm. This procedure is applied recursively until the algorithm determines no further splitting is necessary or each frame is in its own singleton cluster. Top-down clustering is conceptually more complex than bottom-up clustering since we need a second, flat clustering algorithm as a ``subroutine''. However, it has the advantage of being more efficient if we do not generate a complete hierarchy all the way down to individual document leaves. Different algo-

rithms are used in different situations. So again, we suggest further study of machine learning literature.

The following pseudo code exemplifies bottom-up clustering for a generic similarity metric:

```
// Input: A sequence of samples[] (e.g. audio),
// a minimum number of frames for each model segmentlength,
// a similarity metric as a function,
// a treshold
// Output: A sequence labels[]
divisiveclustering(samples, segmentlength,metric(),threshold)
    IF #samples<windowlength:
          return
    run metric() over segments of samples of segmentlength
    determine maximum max for all windows
    IF max>threshold:
          split at sample s with metric() result m
          run divisiveclustering() for samples before s
          run divisiveclustering() for samples after s
          FOREACH segment seg:
                Using metric() compare seg with other segment
                IF results<threshold:
                      give same label to seg and other segment
    ELSE:
          return
divisiveclustering:=labels
```

The reader ask what a suitable implementation for metric is. Again, this depends on the nature of the problem and the underlying data. If an obvious measure cannot be found analyticly (which is the case for most practical multimedia problems), only empirical measurement can quantify the quality of an approach (see also next section on error measurements). Nevertheless, some metric seem to have predominant status in the research community, such as the Minkowski metric and especially the two special cases the Manhattan Distance and the Euclidean Distance.

The Euclidean Distance *d* of two points in n-dimensional space *p* and *q* is defined as:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

The Manhattan Distance is defined as follows:

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n}|p_i-q_i|,$$

In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula. The

Manhattan Distance name alludes to the grid layout of most streets on the island of Manhattan, which causes the shortest path a car could take between two points in the borough to have length equal to the points' distance in taxicab geometry.

In probabilistic algorithms, Entropy (as defined in Chapter XXX) is often used as a metric of homogenity inside a distribution. Derived from that, mutual information is another one that is often used to find out how dependent two random variables *X* and *Y* are on each other.

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right),$$

where $p(x,y)$ is the joint probability distribution function of *X* and *Y*, and $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of *X* and *Y* respectively. Other metrics include KL-Divergence, the Bayesian Information Criterion (see Chapter XXX compression), and others. Again, we recommend studying the literature and related work around the particular problem. Sometimes, distance metrics get invented or rediscovered around an idea because the current ones do not work as well for a specific problem.

Error Measurement and Evaluation

As explained earlier in the chapter and described in Figure 1, the error of a machine learning approach is quantified by comparing the output of the algorithms or systems against ground truth, usually human-annotations. Over the years, different error metrics have established themselves in the multimedia research community. This section will explain some of them. Just like similarity metrics, classification and clustering methods, error metrics should be chosen wisely based on the task that is to be accomplished. A wrongly chosen error metric might lead to a system being optimized towards a wrong goal. Some systems can use standard error metrics, like the following, others require specialized metrics.

The easiest way to measure error, is to classify the output labels into wrong labels and correct labels based on ground truth. The number of wrong labels is then divided by the number of total labels. The quotient can may then be expressed as a percentage. This is a very often and in many cases valid error metric. However, when a more specific distinction than wrong and false is needed, this metric may fail short of being intuitive. Often, several numbers are then used.

Another widely used statistical error metric is precision/recall. It is often used in multimedia information retrieval. Precision is the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search. Recall is defined as the number of

relevant documents retrieved by a search divided by the total number of existing relevant documents (which should have been retrieved). Formally:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Figure 8 depicts the idea.

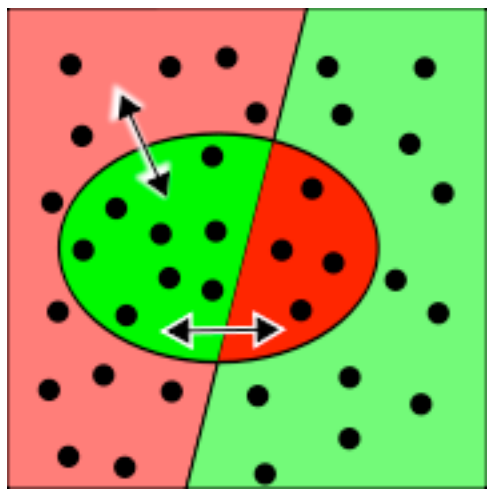


Figure 8. Recall/Precision and F-Measure depend on the outcome of a query in relation to all relevant documents and the non-relevant documents. The oval depicts the outcome of the query, the correct results are green, the wrong results are red. In this picture from [cite Wikipedia], precision is visualized by the horizontal arrow and recall by the diagonal arrow.

Outside information retrieval, the notion is usually used in a generalized manner: Precision for a class c is the *number* of true positives (i.e. the number of items correctly labeled) divided by the total number of elements labeled as belonging to the class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall is defined as the number of true positives divided by the total number of elements that actually belong to the class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). Formally:

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn},$$

with tp, fp, tn, fn, denoting true positive, false positive, true negative, false negative, respectively. A precision 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but does not indicate anything about the number of items from class C that were not labeled correctly). A recall of 1.0 means every item from class C was labeled as belonging to class C (but indicates nothing about how many other items were incorrectly also labeled as belonging to class C). Therefore, the two numbers give a better indication of how the problems is attacked by a certain algorithm. Sometimes, for example, one might want to find only certain items but we should be sure that the answer is correct. This means, we want high precision low recall. When we want many positive results but don't care about them all being correct, high recall and low precision is the goal. Of course, high precision and high recall are almost always desirable in theory but usually hard to achieve in practice. Plotting the fraction of true positives vs. the fraction of false positives is called receiver-operator-characteristics or ROC curve. The so-called Equal-Error-Rate is the point at which both true positive and false positive errors are equal. In general, the algorithm with the lowest EER is most accurate. However, often it might be desirable to choose a different operation point based on precision and recall that is more suitable for the application. For some problems and for comparative reasons, a single number, instead of a curve, might be desirable. Therefore researchers created the so-called F-score that combines precision and recall using the harmonic mean:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

The F-score has often been criticized as counterintuitive because too much information is abstracted away into a single value when in fact sometimes, only the histogram of the classes are a real indication of the error behavior of an algorithm.

As a final example for an application-specific error metric, we present the Word Error Rate (WER) that measures the accuracy of a speech recognition system. The general difficulty of measuring the performance of a speech recognition system is that the recognized word sequence can have a different length from the ground truth word sequence.

The problem is approximated by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment. Word error rate can then be computed as:

$$WER = \frac{S + D + I}{N}$$

with S being the number of substitutions, D is the number of the deletions, I is the number of the insertions, N is the number of words in the reference. One problem with using a simple formula such as the one above, however, is that no account is taken of the effect that different types of errors may have on the human perception and intelligibility of the result, e.g. some errors may be more disruptive than others and some may be corrected more easily than others. As explained earlier, this is a general problem with error metrics and optimizing exclusively by reducing a certain number.

Index Terms

Literature

- Donald Knuth in vol. 3 of The Art of Computer Programming (1973).
- Richard O. Duda, Peter E. Hart and David G. Stork: “Pattern Classification (2nd ed.)”, Wiley Interscience 680 pages ISBN: 0-471-05669-3
- Stuart J Russell, Peter Norvig: “Artificial Intelligence -- A Modern Approach”, Third Edition, PRENTICE HALL 2009.
- Raul Rojas: “Neural Networks A Systematic Introduction”, Springer 1996.
- IH Witten, E Frank: “Data Mining: Practical machine learning tools and techniques”, Morgan Kaufman, 2005.

Research Articles

- When Is "Nearest Neighbor" Meaningful? (1999), by Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, Uri Shaft, In Int. Conf. on Database Theory.
- John C. Platt: “Fast training of support vector machines using sequential minimal optimization”, Advances in kernel methods: support vector learning, pp 185 - 208, MIT Press, 1999
- Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386-408.
- Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press.
- Freund, Y. and Schapiro, R. E. 1998. Large margin classification using the perceptron algorithm. In Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT' 98). ACM Press.

- Widrow, B., Lehr, M.A., "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," Proc. IEEE, vol 78, no 9, pp. 1415-1442, (1990).
- Arthur Earl Bryson, Yu-Chi Ho (1969). Applied optimal control: optimization, estimation, and control. Blaisdell Publishing Company or Xerox College Publishing. pp. 481.
- MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". 1. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297.
- Everitt, B.S. and Hand D.J. "Finite mixture distributions", Chapman & Hall (1981)
- Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.
- Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.
- Schwarz, Gideon E. (1978). "Estimating the dimension of a model". Annals of Statistics 6 (2): 461–464.
- Kullback, S.; Leibler, R.A. (1951). "On Information and Sufficiency". Annals of Mathematical Statistics 22 (1): 79–86.
- L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains", Ann. Math. Statist., vol. 41, no. 1, pp. 164–171, 1970.
- Viterbi AJ (April 1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". IEEE Transactions on Information Theory 13 (2): 260–269
- Ward, Joe H. (1963). "Hierarchical Grouping to Optimize an Objective Function". Journal of the American Statistical Association 58 (301): 236–244.
- Hunt, M.J., 1990: Figures of Merit for Assessing Connected Word Recognisers (Speech Communication, 9, 1990, pp 239-336)
- van Rijsbergen, C. J. (1979). Information Retrieval (2nd ed.). Butterworth.

Exercises

1. Describe how the GMM implementation presented above needs to change for supervized learning.
2. Discuss usability implications for applications that use machine learning based on supervized vs unsupervized training.
3. Implement the backpropagation algorithm and use it train an MLP to learn the XOR function.
4. Use k-NN and GMMs to classify digits (see mm-creole.org)
5. Provide example multimedia tasks of when you think k-NN, GMMs, ANNs, and HMMs might provide good results. Why do you think so? Can you test it?
6. Assuming k-NN gives excellent classification accuracy. Does it still make sense to use other techniques? Why (not)?
7. So-called confidence values provide a means to estimate how certain a classifier is of a decision. This is often desirable when integrating different media together so the different decision can be weighted against each other. Discuss ideas on how to come up with confidence values.
8. Explain strategies for the combination of different media based on GMMs, ANNs and Decision Trees.
9. Explain visually, how an HMM can help in speech recognition.
10. Provide four examples for applications that require low/high precision/recall (respectively)
11. Provide two examples where Word Error Rate does not reflect perceptual error.

12. Explain what happens when models overfit or underrepresent. Discuss strategies to avoid these problems.

Chapter 19: Fundamentals of Audio Content Analysis

In chapter XXX we described the production of sound and its physical properties. In this chapter we will discuss some basic signal processing operations that are common initial steps of many algorithms for audio enhancement and acoustic content analysis.

In this digitized age, all the signals we deal with as computer scientists have been sampled over time (as discussed in section 3.X). In other words, we receive a stream of numbers that are representative of the strength of the signal at certain time points. This representation is therefore also called time-domain, as the x-axis of a graph showing a signal this way would be time. In the frequency domain, every point of the x-axis represents a certain frequency. Visualized as a graph, the diagram would therefore show how the signal varies over frequency. Typically, just like in time domain, the y-axis would show the energy or amplitude of the signal.

A given signal can be converted between the time and frequency domains with a number of mathematical operations. Here we briefly present one of them: the discrete Fourier transform. Chapter XXX on spectral compression presents another.[7]

Discrete Fourier Transform (DFT)

The theory of the Fourier transform goes back to 1822, when Jean-Baptiste-Joseph Fourier found that any function can be described as a sum of sine and cosine functions. Periodic functions need a finite set of sine and cosine functions and aperiodic functions an infinite number of sine and cosine functions. The original Fourier transform assumes continuous functions rather than discrete sequences and an open interval domain. Therefore, in practice, the discretized version of the Fourier transform results in artifacts at the borders of the sequence, which have to be gotten over by windowing strategies. Both the Discrete Fourier Transform as well as the Discrete Cosine Transform are in theory reversible without signal loss but in practice numerical constraints make the transforms often slightly lossy. However, unless noted otherwise, we will assume the transforms to be lossless.

The input to the DFT is a sequence of N real or complex numbers $x_0, ..., x_{N-1}$ which is transformed into a sequence of N complex numbers $X_0,..., X_{N-1}$ by the DFT according to the following equation:

[7] If the concept of frequency space and the math underlying it is not familiar to you, we recommend consulting the literature referenced at the end of the chapter.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn} \qquad k = 0, \ldots, N-1 \tag{8.1}$$

where $e^{\frac{2\pi i}{N}}$ is a primitive N'th root of 1. The inverse transform is given by:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn} \qquad n = 0, \ldots, N-1. \tag{8.2}$$

A direct algorithmic translation of these two equations requires O(N^2) operations (see algorithm 8.1). However, it is possible to execute the same computation with only O(N log N) complexity by factorizing the equation and reusing an intermediate result by applying dynamic programming. The resulting algorithms are called the Fast Fourier Transform and the Fast Cosine Transform. These algorithms are the ones most often implemented into actual codecs. In addition, the DFT and DCT computations are parallelizable (see below) which results in speed gains on manycore architectures.

The following pseudo-code can transform a given signal into frequency domain using the Discrete Fourier Transform. The algorithm is called the Cooley-Tukey Algorithm and it is the most common algorithm for the Fast Fourier Transform.

```
// Input: A number N of samples X. N must be an integer power of two.
  //        s the stride
// Output: The Discrete Fourier Transform of X in the array Y.
FFT(X, N, s)
    IF (N==1) THEN
          Y[0]:=X{0] // Trivial size-1 case: Sample = DC coefficient
    ELSE
          Y[0,...,N/2]:=FFT(X,N/2,2*s) // DFT of (X[0],X[2s},X[4s},...)
          Y[N/2,...,N-1]:=FFT(X[s,...,N-1],N/2,2*s)
                                      // DFT of (X[s],X[s+2s],X[s+4s],...)
    FOR k:=0 TO N/2 // Combine the two halves into one
          t:=Y[k]
          Y[k]:=t+exp(-2*pi*k/N)*Y[k=N/2]
          Y[k+N/2]:=t-exp(-2*pi*k/N)*Y[k=N/2]
FFT := Y
```

Algorithm 8.1: Cooley-Tukey Algorithm for FFT

The elements of the converted signal are called coefficients. The first coefficient corresponds to *sin(π/2)* which is 1. Therefore it is called the DC coefficient (from the electrical engineering interpretation DC = direct current), the rest of the coefficients are called AC coefficients (AC = alternating current).

The Convolution Theorem

Convolution is an operation on two functions f and g that produces a third function. This third function is typically viewed as a modified version of one of the original functions. For example, a function such as an image, a video, a set of measurement samples, or an audio track may be modified by a second function, usually called a filter. We introduced the notion of frequency space in Chapter XXX (advanced compression). Here we introduce a fundamental correspondence between the time space and the frequency space: the Convolution theorem. It states that a convolution filter in one domain (e.g. time) equals point-wise multiplication in the other domain (e.g. frequency). In multimedia computing, this is the most important fact about convolutions.

Let f and g be two functions and $f * g$ their convolution. (Note that * denotes convolution in this context, and not multiplication). Let $\mathcal{F}$ denote the Fourier transform, so $\mathcal{F}\{f\}$ and $\mathcal{F}\{g\}$ are the Fourier transforms of f and g, respectively. Then

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \quad (8.3)$$

where the dot denotes point-wise multiplication. The other direction is also true:

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\} \quad (8.4)$$

By applying the inverse Fourier transform, one can write:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\} \quad (8.5)$$

Mathematically speaking (8.3) – (8.5) are only valid for the canonical form of the Fourier transform. In practice, however the transform may be normalized in other ways, in which case constant scaling factors may appear. Most importantly, the theorem allows us to design convolution filters in either time or frequency space whichever is more convenient for the concrete problem. This is exactly what is done in practice and it is also why some of the frequently used filters presented in this chapter have their “home” in frequency space and some others seem to be mostly applied in time space.

Linear Filters

A very common type of filter useable in many situations and in many domains is the linear filter. The linear filter applies a linear operator to a time-varying input signal.

Definition 8.1: Linear Operator

Let V and W be vector spaces over the same field K. A function $f : V \to W$ is said to be a *linear operator* if for any two vectors x and y in V and any scalar a in K, the following two conditions are satisfied

1) Additivity: $f(x+y) = f(x) + f(y)$

2) Homogenity (of degree 1): $f(ax) = af(x)$

It follows from the conditions that $f(0)=0$.

These mathematical properties have several advantages in real-world applications. For example, because of the homogenity rule, it does not matter wether a linear filter is applied before or after amplification of a signal. The same applies for the additive mixture of two signals: The additivity makes sure that it does not matter whether the filter is first applied to the each of the mixture components before the mix or after the mix to the added components. This is especially useful to generalize a 1-dimensional linear filter to more than one dimension: One can just apply the filter to each dimension individually. Because of these properties, many real-world circuits and filter formulas aim to be linear. Distortions are often caused by deviations from the linearity assumption of a real-world equipment, such as amplifiers (which are essentially multipliers of the signal in time space).

Linear filters are further divided into two classes: infinite impulse response (IIR) and finite impulse response (FIR) filters. FIR filters (which may only be implemented on a discrete time scale) can be described as a weighted sum of delayed inputs. If the input becomes zero at any time, then the output will eventually become zero as well, as soon as enough time has passed so that all the delayed inputs are zero, too. Therefore, the impulse response lasts only a *finite* time. In an IIR filter the set of time where the output is non-zero will be unbounded; the filter's energy will decay but will be ever present.

The mathematical discipline behind filter design is called linear time-invariant (LIT) system theory. The following section discusses some commonly used filters.

Common Linear Filters

Linear filters are often used to eliminate unwanted frequencies from an input signal or to select or enhance a set of desired frequencies from among many others. As the reader might already have inferred, these operations are among the most important. Common types of linear filters include the low-pass filter, which passes frequencies below a threshold; a high pass filter, which

passes frequencies above a threshold; and a band-pass filter, which passes frequencies of a band, between two frequency thresholds. Similarly, pre-emphasis amplifies a certain band (see Chapter XXX). A band-stop filter passes frequencies except a limited range, i.e. it is the reverse of the band-pass filter. A special, often used, band-stop filter is the so-called notch-filter. Notch-filters have a particularly narrow stop band and are often used in audio equipment, one example being the anti-humm filter that filters electro-wire-induced humming between 59 and 61 Hz or 49 and 51Hz (depending on the country's electrical specifications). The following example implements a typical band-pass filter:

```
// Input: 4096 audiosamples s[] sampled at 44100 Hz,
  // lower boundary frequency lf, upper boundary frequency uf,
  // an FFT function, an inverse FFT function (see Chapter XXX)
// Output: A filtered audio signal
bandpass(s[],lf, uf , fft, ifft)
    fs[][]:=FFT(s) // convert to frequency space with blocksize 4096,
                   // 2nd index 0-real value, 2nd index 1 = complex value
    FOR i:=0 TO 2048: // exploit symmetry
          IF ((44100/4096)*i<lf) OR ((44100/4096)*i>hf): // bandpass
                fs[i][0]:=0
                fs[i][1]:=0
                fs[4096-i-1][0]:=0
                fs[4096-i-1][1]:=0
    s:=iFFT(fs) // back to time space
bandpass := s[]
```

Algorithm 8.2: Band-pass Filter

Equalizers use combinations of band-pass filters to extract individual bands that can then be amplified or suppressed according to user settings. Equalizers are part of virtually any home audio system and can be used to fine-tune the frequency spectrum of an audio signal, e.g. more bass can be added, and speech can be made more intelligible against background noise. The wrong settings, however, can degrade audio quality. Since the individual bands can be easily visualized using sliders, the filter is often called "graphic equalizer". Figure 1 shows a graphical equalizer that is implemented by simulating a 10-band octave filter array conforming to ISO Recommendation 266. The filters use a Gaussian curve which allows the equalizer to operate smoothly across the different bands. The center frequency occurs at the top of the Gaussian curve and is the frequency most affected by equalization. It is often notated as f_c and is measured in Hz. The 11th slider on the left regulates pre-amplification of the signal before it enters the filter bank.

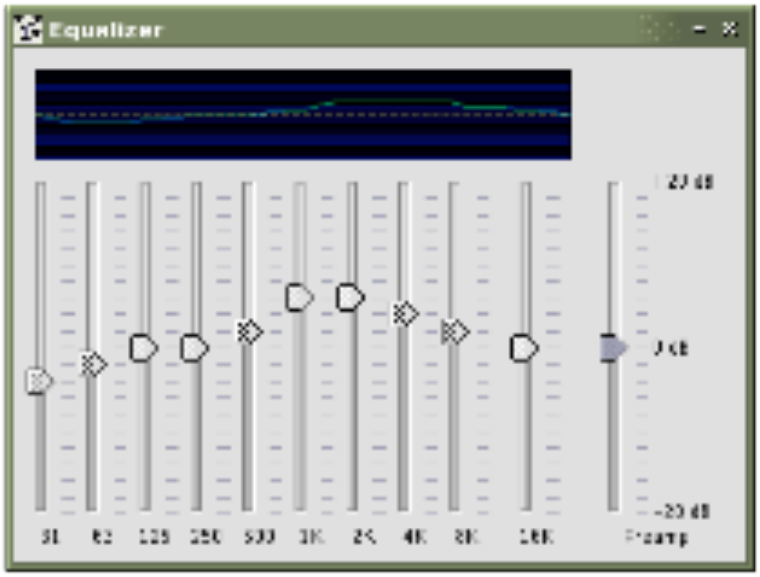


Figure 1. A graphic equalizer in a typical setting used for enhancing the intelligibility of speech. The numbers on the bottom show the center filter frequency in each band and the sliders manipulate the energy in each band.

Non-Linear Filters

Not all filters are linear. In fact, most filters are not -- but of course that does not mean that they are not useful.

One important non-linear filter is dynamic range compression (DRC). This filter often results in surprising perceptual enhancements of the audio signal. The analog process of DRC, sometimes simply called compression, is not to be confused with audio compression as discussed in Chapter XXX, even though DRC can lead to better data compression rates. DRC is a process that reduces the dynamic range of an audio signal, i.e. it narrows the difference between high and low time levels. DRC is applied by running an audio signal through a dedicated electronic hardware unit or, nowadays, mostly through software. In the context of audio production, the hardware unit is often simply referred to as a "compressor". A DRC is a volume control filter. Two types of compressors exist: Downward compressors work by reducing loud sounds over a certain time threshold while lower amplitude sounds remain untreated. Upward compressors amplify sounds below a threshold while loud signals remain unchanged. In both cases, the dynamic range of the audio signal is reduced. DRCs are often used for aesthetic reasons or sometimes to deal with technical limitations of audio equipment, e.g. to avoid clipping.

Dynamic range compression often improves audibility of audio in noisy environments when background noise overpowers quiet sounds. In these cases, compressors are tuned so that they reduce the level of the loud sounds but not the quiet sounds, whose level can be raised to a point where quiet sounds are audible without loud sounds being too loud. Note that analog compressors always decrease the signal to noise ratio. While this is sometimes acceptable for human perception it is often a problem for machine learning algorithms. Figure 2 shows the functionality of a typical compressor.

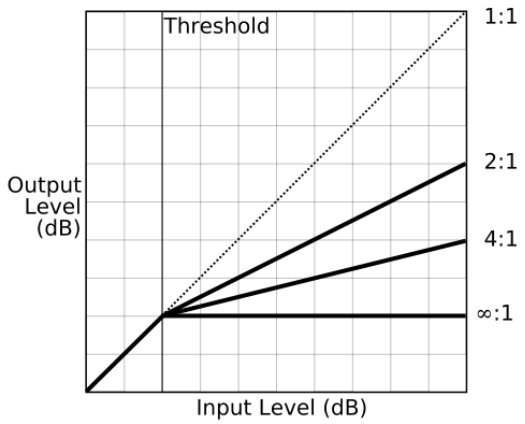


Figure 2. Relationship between input level, output level, and gain reduction in a set of example downward dynamic range compressors. This particular type of DRC amplifies the signal below a threshold amplitude and then dampens the signal above the threshold amplitude.

Building compressors, especially as analog circuits, can be complicated because they require some manual fine tuning. For example, usually, the bend in the response curve around the threshold is smoothed to be a rounded edge rather than a sharp angle as in Figure 2. It is often desirable for a compressor to provide control over how quickly it acts as the volume pattern of the source material that is modified by the compressor may change the character of the signal quite noticeable. Some compressors therefore allow us to define an attack and a release phase similar to the attack and release phases of a synthesizer (see Section 3.X). Another difference is between peak sensing and RMS-energy sensing (compare Chapter XXX speech compression). A peak sensing compressor responds to the level of the input signal instantaneously while an RMS-energy compressor responds to the energy of the signal over a small time window. The second type of compressor usually resembles perception of volume more closely. Other factors include the way a compressor reacts to stereo signals or if it reacts to different sub-bands differently (multi-band compressor). In the digital age, creating compressors no longer involves fiddling

around with complicated control circuits, but tuning might still be necessary. Exercise # asks you to design a simple software compressor.

Downward and upward DRCs might be combined using two or more different thresholds to form a variable-gain amplifier. Figure 3 shows a typical signal before and after the application of a variable-gain amplifier.

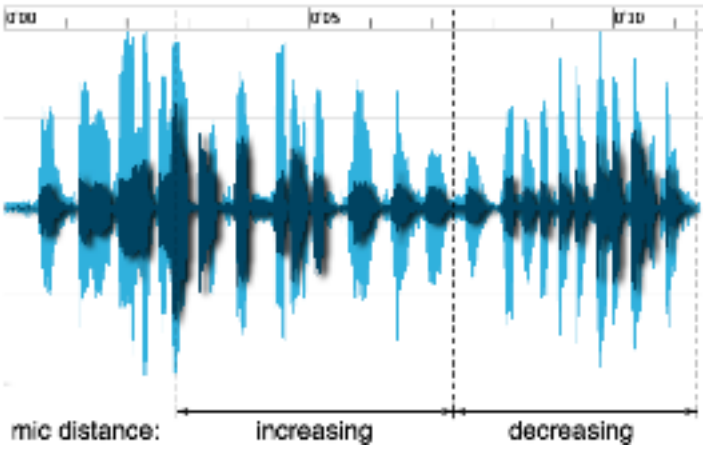


Figure 3. An example application of variable-gain amplifier. When the speaker turns away from the microphone, the audio gain goes down and with the mouth approaching the microphone the gain raises again (darker signal). Using DRC, the overall gain is higher and the microphone distance differences are leveled out more effectively (lighter signal).

Multiband variable-gain amplification is often designed to approximate the Mel or Bark curves (see Chapter XXX). This can be used to normalize sound files in a collection that have been recorded at different levels to about the same audible level. This function is often part of both software and hardware MP3 players.

Another set of very important non-linear filters are resampling (subsampling and supersampling) filters. *Subsampling* (or *downsampling*) is the process of reducing the sampling rate of a signal. This is regularly done, for example, when a smaller thumbnail of an image is created or when compact disc audio at 44,100Hz is downsampled to 22,050Hz before broadcasting over FM radio. Since downsampling reduces the sampling rate, one must be careful to make sure the Nyquist sampling theorem criterion is maintained otherwise aliasing will occur (see Chapter XXX). If a signal has been recorded at a much higher sampling rate it is very likely that the original signal will contain frequencies higher than can be represented by the downsampled signal. To ensure that the sampling theorem is satisfied, a low-pass filter has to be used as an anti-

aliasing filter to reduce the bandwidth of the signal before the signal is downsampled. When downsampling by integer factors (n = 2,3,4, etc.) all one has to do is apply a low-pass filter to the signal and then pick every nth sample from the low-pass filtered samples. When downsampling to a rational fraction m/n, the signal should be supersampled by factor m first, so that it can be downsampled by factor n in the second step.

Supersampling (or *upsampling*) is a bit more tricky. Supersampling is the process of increasing the sampling rate of a signal. For instance, upsampling raster images such as photographs means increasing the resolution of the image. Of course, objectively information cannot really be gained by algorithmic upsampling because the supersampled signal satisfies the Nyquist sampling theorem if the original signal does. So the basic supersampling algorithm is called zero-stuffing: Insert as many zeros as needed between the two original samples and then apply a low-pass filter with a threshold set at the Nyquist limit frequency of the original signal. However, more elaborate algorithms try to model the underlying continuous signal from the original samples to then sample it again at a higher sampling rate, trying to convert alias into real information. These filters are often called *interpolation filters*.

The most basic interpolation filter is the constant interpolation filter: Repeat each value n times (where n is the upsampling factor) and then apply a low-pass filter. *Linear interpolation* is almost as basic: Instead of duplicating each sample, assume a linear function between each of the original samples and use that to create as many new samples between the original samples as required by the upscaling factor. Generally, linear interpolation takes two data points, (xa,ya) and (xb,yb), and the interpolant is given by:

$$y = y_a + (y_b - y_a)\frac{(x - x_a)}{(x_b - x_a)}$$

Because sound waves are generally sinusoidal rather than linear, these simple interpolation techniques will create (sometimes audible) artifacts. Therefore, polynomial and other nonlinear functions are often used. *Polynomial interpolation* is a generalization of linear interpolation. Generally, if with n data points, there is exactly one polynomial of degree at most $n-1$ going through all the data points. However, calculating the interpolating polynomial is computationally expensive compared to linear interpolation. Suppose that the interpolation polynomial is in the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

We want $p(x)$ to interpolate all data points. In other words:

$$p(x_i) = y_i \qquad \text{for all } i \in \{0, 1, \ldots, n\}.$$

By substituting equation p(x) we get a system of linear equations in the coefficients a_k, which in matrix form reads:

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

This system of equations can be solved by Gaussian elimination. Figure 4 compares the three approaches for a sample signal.

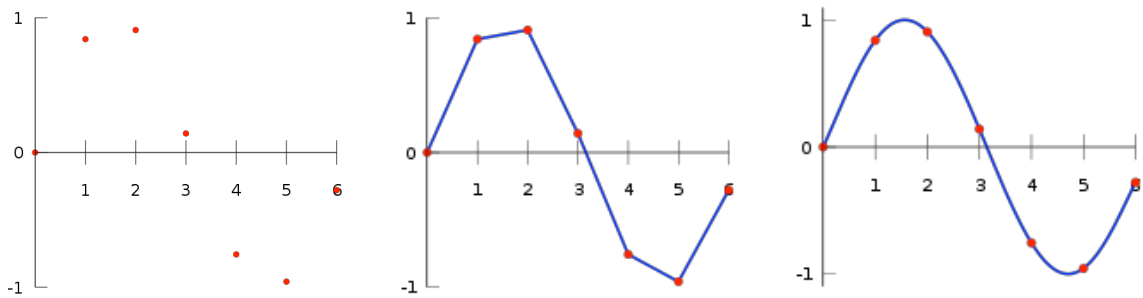


Figure 4. Left: original signal, middle: linear interpolation, right: polynomial interpolation. Even though the polynomial interpolation looks similar to what the original signal might have looked like, there is an infinite number of other possibilities how the original (continuous) signal might have looked like.

Because of the computational complexity of polynomial interpolations, *spline interpolation*, which implements piecewise polynomial interpolation with low-degree polynomials is often used to interpolate vector graphics efficiently. In audio processing, *polyphase filters* are more common; they help avoid aliasing and further distorting the signal by up- or downsampling. Polyphase filtering divides the signal into different sub-bands which are treated independently. Using the critical sub-bands (see Chapter XXX) for example, can help reduce the effects of human audible aliases.

Filter by Example

Especially for noise reduction it is often desirable to be able to eliminate a very specific part of the signal that is not necessarily constant in frequency range or might not be sinusoid. For example, when digitizing a CD from an old vinyl record, the scratches of the pickup might cover a range of frequencies so large that using a band-stop filter might distort the signal significantly. Therefore, a common technique to eliminate such noise is to perform the digitization normally, create a noise fingerprint based on a region of otherwise silent parts of the source, and then apply

a technique called *spectral subtraction*. Spectral subtraction works under the assumption that noise has been added to the signal and therefore literally subtracts the noise fingerprint from the actual signal in frequency space. In other words, both the noise fingerprint and the signal are converted to frequency space and then the energy values of the noise fingerprint are subtracted from the signal. Often the noise fingerprint is pre-amplified or reduced depending on the situation. The signal might then be converted back to time space. The popularity of spectral subtraction is due to its relative simplicity, ease of implementation, and effectiveness especially in cases where complicated but constant noise-patterns make filtering with band-stop filters cumbersome. Figure 5 shows the effect of spectral subtraction. The downside of spectral subtraction is, of course, that it can seriously distort the signal and is not applicable in all cases.

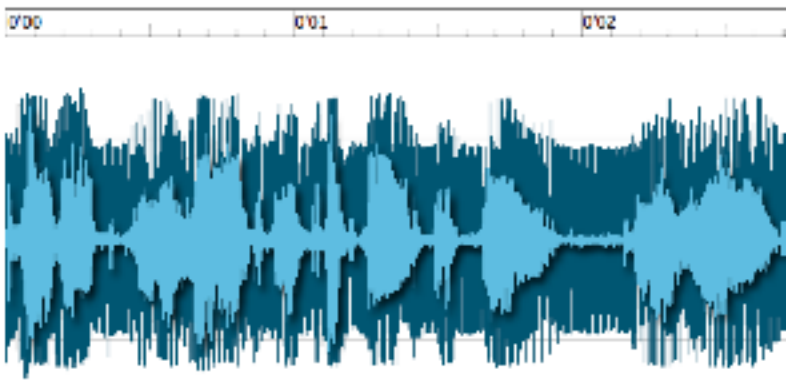


Figure 5. Three seconds of a speech signal (time space) with a 100Hz-sine-like humming before (dark) and after spectral subtraction (light). Humming is a frequent audio distortion when recording in situations that require long wires.

Another, very commonly used approach to reducing noise based on an example signal is the *Wiener Filter*, proposed by Norbert Wiener in 1949. Its purpose is to reduce the amount of noise present in a signal by comparison with an estimation of the desired noiseless signal. The other filters discussed in this chapter are designed to produce a desired frequency or amplitude response. However, the Wiener filter takes a different approach: One is assumed to have knowledge of the spectral properties of the original signal and the noise. Wiener filters are characterized by the assumption that the signal and (additive) noise are a stationary linear stochastic process with known spectral characteristics and known autocorrelation and crosscorrelation. Now the Wiener Filter seeks the linear filter whose output would come as close to the original signal as possible using the least mean-square error as optimization criterion. The input to the Wiener filter is assumed to be a signal $s(t)$ corrupted by additive noise n(t). The output $s^{\wedge}(t)$ is calculated by means of a filter $g(t)$ using the following convolution:

$$\hat{s}(t) = g(t) * [s(t) + n(t)]$$

The error is consequently defined as:

$$e(t)=s(t)-s\hat{}(t)$$

which can be formulated as the quadratic error:

$$e^2(t)=s^2(t)-2s(t)*s\hat{}(t)+s\hat{}^2(t)$$

By applying the convolution theorem and other math, one can formulate the underlying optimization problem in frequency space as follows:

$$G(s(t))=|X(s(t))|^2/((|X(s(t))|)^2+(|N(s(t))|)^2) \quad (8.6)$$

Where $X(s(t))$ is the power spectrum of the signal s(t) and $N(s(t))$ is the power spectrum of the noise and $G(s(t))$ is the frequency-space Wiener filter. If the shape of $N(s(t))$ is known, an arbitrary optimization algorithm may be used. In practice the main problem is that accurate estimates of $N(s(t))$ do not exist. Therefore, often $N(s(t))$ is simply assumed constant at the level of the signal to noise ratio. The following pseudo-code implements the calculation of a simple Wiener Filter this way:

```
// Input: An signal S[], an SNR snr, and a fingerprint P[]
// Needs: FFT function fft(), inverse FFT function ifft()
// Output: Wiener Filter F[]
Wienerfilter(S,snr)
    const:=1/(snr*snr)
    fft_F:=[][] // real part in fft_F[][0], imaginary in fft_F[][1]
    fft_S[][]=fft(S) // real part in fft_S[][0], imaginary in fft_S[][1]
    fft_P[][]=fft(P) // real part in fft_P[][0], imaginary in fft_P[][1]
    FOR i:=0 to length(fft_P):
          denominator:=fft_P[i][0]*fft_P[i][0]+fft_P[i][1]*fft_P[i][1]+const
          fft_F[i][0]:=fft_P[i][0]*fft_S[i][0]+fft_P[i][1]*fft_S[i][1]
          fft_F[i][1]:=fft_P[i][0]*fft_S[i][1]-fft_P[i][1]*fft_S[i][0]
          fft_F[i][0]:=fft_F[i][0]/denominator
          fft_F[i][1]:=fft_F[i][1]/denominator
    F:=ifft(fft_F)
    WienerFilter:=F
```

Algorithm 8.3: Wiener Filter

Graphical Filters

Some filters can be useful in practice even without changing the input signal. One of the most frequently used examples of such a filter is the *VU meter*. A VU meter is often included in audio equipment to display the signal level in so-called Volume Units (which are proportional to the energy of the signal). VU meters try to reflect the perceived loudness of the sound by measuring the volume slowly, averaging out peaks and troughs of short duration. VU meters are standardized by IEC 268-10:1974. Figure 6 shows a typical VU meter that displays both the peak signal and the average signal level. It also counts clippings (when signals more than 98% of the maximum allowed range). The average gain level is measured by calculating the root-mean-square value of a time window of 250ms. The value ages with the last three measurements. The ideal recording maximizes the average signal without causing overrun.

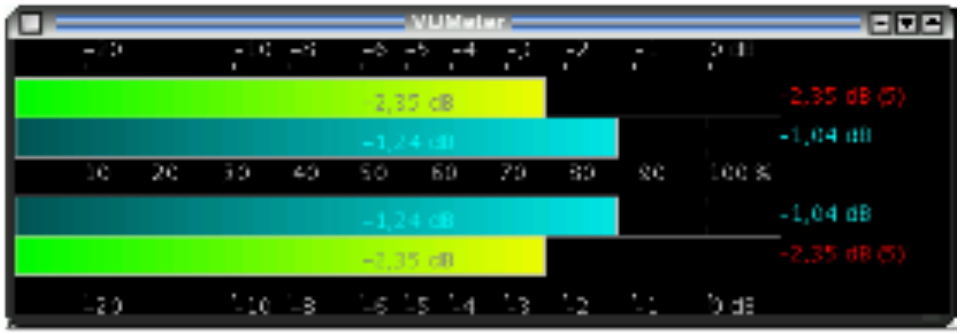


Figure 6: A screenshot of a typical (software) VU meter. This view shows a VU meter in stereo mode with a mono signal fed in. The inner bars show the average gain while the outer bars show the peak gain. Clippings are counted and displayed in red next to the peak gain meter.

Features for Audio Analysis

While machine learning algorithms (as described in Chapter XXX) can usually be used unchanged for different types of input data. The features are that are used as input for machine learning are different because the sensor output for audio is different from vision for example (microphone vs. camera). This section describes commonly used features.

Energy/Intensity

The most frequently used feature and at the same time most basic audio features is energy often also called intensity. As already described in Chapter XXX. The most common form of energy features for content analysis is obtained by taking the root-mean-square x_{rms} if the n samples x_i :

$$x_{\mathrm{rms}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2} = \sqrt{\frac{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}{n}}.$$

Pitch

We already described the calculation of pitch in Chapter XXX (speech compression). While pitch alone is rarely used directly for content analysis (except if the goal is to extract pitch), pitch is used as a basis for many features and therefore very important. In speech, detecting voiced and unvoiced regions has high value because unpitched regions are more in many domains more likely to contain noise. The ratio between pitched and unpitched regions (in time) is called Harmonicity-to-Noise Ratio or HNR. HNR can be used as one feature to classify music instruments. It can also be used for detecting a speaker's age as HNR is age dependent as older speakers have a "rougher" voice, i.e. less harmonicity. Often HNR is approximated using a threshold on the zero-crossing rate of the signal (i.e. the number of times the signal changes sign).

Long-term Average Spectrum (LTAS)

The Long-term Average Spectrum is often used as a feature in speaker identification. It can also be used to classify different recording environments. The LTAS is not a single value but a feature vector. In order to obtain LTAS one calculates the FFT of a signal (see Chapter XXX) and averages the energies in each band over a reasonable amount of time (typical a couple of seconds).

Formants

Formants are the distinguishing or meaningful frequency components of human speech and also of human singing. The information that humans require to distinguish between vowels can be represented purely quantitatively by the frequency content of the vowel sounds. The formant with the lowest frequency is called f_1, the second f_2, the third f_3, and so on. Usually, f_1 and f_2, are enough to disambiguate a vowel. These two formants determine the quality of vowels in terms of the open/close and front/back dimensions. Thus the first formant f_1 has a higher frequency for an open vowel (such as [a]) and a lower frequency for a close vowel (such as [i] or [u]); and the second formant f_2 has a higher frequency for a front vowel (such as [i]) and a lower frequency for a back vowel (such as [u]). Vowels will almost always have four or more distinguishable formants; sometimes there are more than six. Formants are often measured manually as an amplitude peak in the frequency spectrum of the sound, using a spectrogram. In music processing,

formants refer to a peak in the sound envelope and/or to a resonance in sound sources, notably musical instruments as well as that of sound chambers.

Different algorithms are available to track formants. A common way of doing it is to resample the audio signal to a sampling frequency to twice the value of the maximum expected formant (which varies by sex and age, for a young child it could be up to 5500 Hz). Then, LPC coefficients are calculated (see Chapter XXX) and searched for local maxima close to the expected frequency ranges of the formants. Figure 7 shows a visualization of some of the features discussed thus far, including formants.

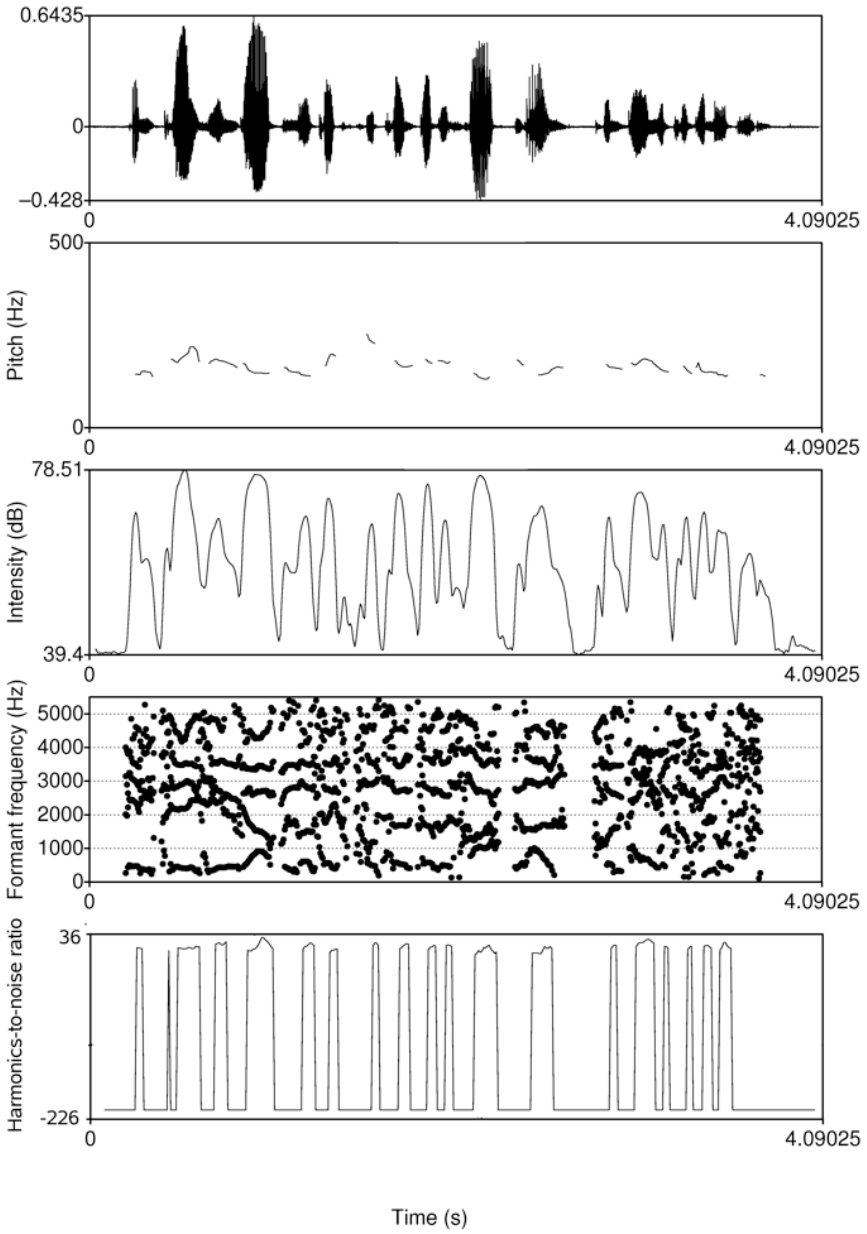


Figure 7. Visualization of some of the features described in this section. (From IEEE TASLP article, co-authored by me)

Linear Prediction Coefficients (LPC)

The LP coefficients originally developed for speech compression and therefore discussed in Chapter XXX are an often used feature for various speech tasks, such as speech recognition. The coefficients as well as the residual capture the characteristics of different aspects of the signal.

LPCs are usually computed on small windows of the signal, for example, 10-30ms. A small window like that is usually called a frame. A frame is the smallest unsplittable unit of analysis.

More frequently used than LP coefficients, however, are the MFC coefficients which are so important that we describe them in their own section.

Mel Frequency Cepstral Coefficients (MFCCs)

Human perception is different between different media. However, it is not clear if the computer should simulate this distinction. Clearly, when machine learning accurately models the human brain, then it would make sense, for example to quantize sensor output logarithmically (see Chapter XXX). However, since current machine learning algorithms are mostly statistical methods, it is not clear if this is beneficial. Nevertheless, some features, such as the Mel Frequency Cepstral Coefficients do incorporate human perceptual properties. In the case of it works remarkably well as MFCCs can be seen as the most frequently used features for any speech analysis tasks, such as speech recognition, speaker identification or language identification. Take a look at Figure 8, which shows the steps involved in the generation of MFCC features as a diagram.

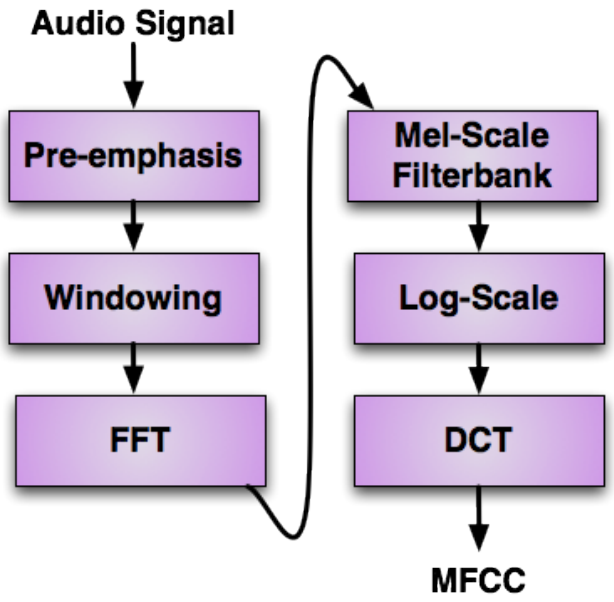


Figure 8. The steps involved in calculating MFCCs.

MFCCs are commonly derived as follows:

- Perform a pre-emphasis of the signal
- Take the Fourier Transform of a windowed excerpt of a signal, typically 10-30ms.
- Map the powers of the spectrum obtained above onto the Mel scale (see Chapter XXX) using triangular overlapping windows.
- Take the logarithm of the powers at each of the Mel frequencies (see Chapter XXX).
- Take the Discrete Cosine Transform (see Chapter XXX0 of the list of Mel log powers, as if it were a signal

The following pseudo-code implements the triangulation-shaped Mel-filter:

```
// Input: samplingrate sr, number of Fourier bins nf, number of Mel-scale co-
efficients nm
// Output: A Mel-scale filterbank Matrix M for convolution with an audio sig-
nal.
melfilter(sr, nf, nm)
      nyq := sr/2
      nyq_mel := 2595 * log10(1 + nyq/700.)
      M[nf][nf] := new_zero_matrix()
      FOR i:=0 TO nf-1:
            f := i * nyq / nf
            f_mel := 2595 * log_10(1 + f/700.)
            m_idx := f_mel/nyq_mel*nm
            j=floor(m_idx)
            M(j+1,i) = m_idx-j
            M(j+0,i) = 1.0-m_idx+j
melfilter := M
```

The last DCT step might seem odd to the reader since it means to transform a frequency-space based signal into frequency space. In fact, that's exactly what's being done and the original creators found this so interesting that they named it "cepstrum" as a word-play that exchanges a couple of characters from the word "spectrum". The reason for doing this is that the values obtained in Mel-frequency spectrum is still quite (depending on the window of the FFT). So to further abstract the signal into lower dimensionality, the DCT is used to decorrelate and then reduce the information further, eg. to 12 dimensions (a typical value for speech recognition).

Often, acoustic analysis is performed using "12 dimensional MFCC with delta and delta-deltas". This means, a 12 dimensional DCT is performed as final step and then the differences between the 12 values (deltas) are computed and the differences between the deltas are also computed (delta-delta). This is to approximate the MFCCs and their first and second derivate. For speaker recognition and diarization (see below), often 19 or 22 dimensional MFCCs are computed as the higher dimensions are said to contain more speaker and channel information. The implementation of MFCC calculation is left as an exercise.

Typical Audio Analysis Tasks

After explaining fundamental methods needed for multimedia content analysis and audio processing, the remaining of this chapter explores some of the building blocks and applications of acoustic content analysis. We will provide a short overview of how typical speech and music analysis systems work by describing on a high level which signal processing and machine learning techniques typically are used. Note that all of the systems presented here will work as presented. However, to achieve high accuracies they require a significant amount of engineering. To go beyond a certain accuracy, research is needed which might redefine how typical systems work in the future (thus potentially making our descriptions obsolete).

Speech Activity Detection

A very fundamental task in the analysis of an audio signal is to separate human-uttered language from the remaining signal. This function is needed in almost any task that works with language, including speech compression. Typically, however, the methods used in speech compression that detect speech regions in and audio stream are by far not as accurate. Some of them were discussed in Chapter XXX (speech compression), including energy thresholding, and voiced/unvoiced detection. The biggest challenge with any of the basic methods is the distinction between speech and noise of similar characteristics. Therefore, an approach that usually works better is to build a classifier and train models based on audio files that contain speech in a similar characteristics as the speech to be detected and other models on noise in similar characteristics as the noise to be distinguished from speech. A simple approach, is to train two sets of Gaussian Mixture Models based on 12-dimensional MFCC features that include energy and delta, and delta-delta coefficients. The decision is usually made on a frame-by frame basis. A HMM in combination with the Viterbi algorithm (see Chapter XXX) can then be used to make an optimal decision for a larger region of the audio stream or file. By training many hours of data into the models current speech activity detectors (often also referred to as speech/non-speech detectors) obtain accuracies of up to 98% when the characteristics of the training data matches that of the evaluation data, e.g. models trained on broadcast TV and applied to broadcast TV or models trained on telephone speech and used in similar phones. The development of a simple model-based speech activity detector is left as an exercise.

Large Vocabulary Automatic Speech Recognition

Speech recognition engines are usually quite large systems. While small-vocabulary speech recognition is used for command and control, e.g. for telephone centers, and pretty much a standard

product in industry, state-of-the-art speech recognition engines for large sets of vocabulary and conversational speech contain the work of many many researchers. As a consequence, complete speech recognizers barely exists in universities. Universities usually only deal with certain aspects of the task. It is a domain of companies and research institutes.For that reason and since large vocabulary automatic speech recognition (LVASR) is the most important field in speech processing research, we provide here a rough overview of the functionality of an automatic speech recognizer.

- Feature Extraction

Speech recognition usually starts with several layers of signal processing (e.g., pre-empahsis, windowing, short-term spectral analysis and filtering, and so on). The predominant features used for speech recognition are MFCCs (see above). Although for special purposes, such as high-noise ASR, other features have been designed such as the so-called PLP and RASTA features which are described in research papers the can be found under references.

- Speech Activity Detection

As with most other speech tasks, the first step is a speech activity detection as described above.

- Feature Normalization

After features have been extracted and non-speech is eliminated, the next goal is to try to make the features invariant to anything but the spoken words. Remember that MFCCs are used for various acoustic content analysis tasks. Ideally, we want to eliminate any statistical dependency on the speaker or the channel (microphone, room reverberation). Therefore many techniques exist to normalize features, some are very basic, like Gaussianization, some of them are pretty advanced like Vocal Tract Length Normalization (VTLN). Gaussianization takes a set of audio features and normalizes them so that the histogram of the values forms roughly a Gaussian. This is similar to image histogram equalization (see Chapter XXX) except the target function is a Gaussian rather than a flat distribution. VTLN is further described in the references to this Chapter.

- Recognition

Now that the audio is filtered so it hopefully contains only speech and features that are invariant to everything but the actual spoken words one uses a classifier, such as a GMMs to compare the spoken words on different levels (using different window length) to the recorded and annotated words in our acoustic models. Usually, a large number of Gaussians that are used in combination

to generate likelihoods for particular speech sounds in context are used. The parameters of this acoustic model are then altered further for testing by incorporating one of several related methods for adaptation, for instance Maximum Likelihood Linear Regression (MLLR) (see references). The models are often then trained in a new pass of discriminant learning using techniques like Minimum Phone Error training. In the end, the idea is to recognize "a" by comparing it to all instances of "a" stored in our acoustic model. It is considered to be an "a" if it is very close to all the other "a" and not so close to any other acoustic element, such as "e" or "o" . Using different window length one can compare on sub-phoneme, phoneme, syllable and word level.

- Decoder

Once small-scale recognition (e.g. phonemes, syllables, etc..) is done, the next goal is to glue the pieces together using a so-called language model. The entire acoustic likelihood estimation subsystem is used in combination with a language model probability estimation, which has been trained in a supervised fashion on a large number of words; additionally, there are usually multiple sources of word prediction information (such as large quantities of written text and smaller amounts of transcribed spoken words). Usually HMMs are used to model phoneme and word sequences. For each acoustic instance the recognizer usually outputs a set of alternatives with probabilities, which are used as observations in the HMM. The language model chooses the most likely combination of phonemes, syllables and words, according to the recognizer output. A very hard problem is to handle words that are not part of the language model and usually results in high error rates as surrounding words are also affected.

- Textual Postprocessing

Once decoding is done processing has to be done that may take into account prosody, speech pauses, and other hints to detect sentence boundaries so that punctuation decisions can be made. Also, named entities should be detected so that capitalization works.

This description of automatic speech recognition only conveys the general idea of this class of systems. It shows, however, how the different content analysis and machine learning techniques work together (see Chapter XXX). Speech recognition systems usually contain many signal processing, classification, temporal modeling, and other content analysis tricks that work together.

Speaker Recognition

Speaker recognition is the general term used for acoustic content analysis tasks where the identity of the speaker is to be found automatically by the system. This task has various real-world applications, including forensic analysis, door opening systems, and multimedia retrieval. Depending on the application, there are various “guises” of speaker recognition. Perhaps the most natural form is that of speaker identification, which is to identify the identity of the speaker from a spoken utterance, given the set of possible speakers of that utterance. However, in practical situations it hardly ever occurs that the set of possible speakers is limited, rather, usually there needs to be some verification that the speaker is actually one of the set. Of course, this has to be done after we detected that the audio segment in question is actually speech. Allowing for the possibility of out-of-set speakers is termed open-set speaker identification, and requires that internal similarity scores have some form of “absolute” meaning so that a score can be thresholded, and a hypothesized speaker can be rejected if the score is too low. This capability of rejecting an unknown speaker is so important, that it has been the main focus in speaker recognition methods and its performance evaluation. For non-discriminative modeling, the open-set speaker recognition problem can be generalized to the speaker detection task, where the task is to decide whether or not a given speech segment is spoken by a target speaker. As this general task is at the basis of many different application scenarios, we will use the speaker detection task (equivalent to one speaker open-set identification) as the prototype task in this description.

- Universal Background Model

In order to cope with the open set problem, a Universal Background Model (UBM) is used that represents the speech of "all" possible speakers. It is essentially a GMM consisting of many Gaussians, typical figures are 512–2048 (traditionally, the number of Gaussians are chosen as powers of 2). A UBM is used as denominator in determining a likelihood ratio, representing the likelihood of the “alternative speaker” in speaker detection, i.e. it is used to normalize the score by determining wether the likelihood score obtained by the GMM is typical of a match or might be equally found in two random similar but different speakers. Of course there is no way to represent all possible speakers, yet thousands of speakers are usually used to train the UBM.

In addition to normalizing the score, UBMs are often used as the starting point for modelling a specific speaker, which can be found by adapting the UBM using limited amounts of speech

from a specific speaker. It is often the displacement of the centers of the Gaussians that are used to completely characterize a speaker.

- General Architecture

There is a specific training, or enrollment, phase of a speaker, and a testing phase, we can differentiate between the common parts and the training/testing specific parts of the architecture. The common processing steps for a given speech segment are:

1. Speech Activity Detection
2. Feature extraction, usually MFCCs
3. UBM index generation: This step computes the contribution to the UBM likelihood of every Gaussian component, for every frame of the speech segment. Then the indices of the N top-most contributors are extracted, typically N = 5 Gaussians are used. The idea is that these five are enough to compute the likelihood of the frame accurately.
4. Supervector generation: Using the top-N Gaussians per frame in calculation, the means of the UBM can be adapted to maximize the a-posteriori likelihood of the speech segment (so-called MAP adaptation). The shift in means can be said to represent the speaker of the speech segment. A per-dimension scaling of this displacement using the prior and variance parameters of the UBM and concatenation of the scaled displacement vectors into a so-called supervectors allows a geometric interpretation of this space. A speech utterance is represented as a point in this space, and when points lie close together we consider it more likely that the speech was uttered by the same speaker.

The steps specific to training are:

Model generation: There are two distinct classes of modelling used in speaker recognition: generative and discriminative. For a generative model, the MAP-adapted GMM is the model—the important parameters are the (unscaled) means of the Gaussians. Alternatively, a discriminative model can be formed by using a Support Vector Machine (SVM). Additional to the target speaker, for which the model is to be trained, many non-target (i.e., “background”) speakers are used to compare the target speaker to. As described in Chapter XXX, the SVM tries to maximize the margin between the target speaker and the background speakers. That is, it tries to position a hyperplane in supervector space which has a maximum distance from the target speaker. The SVM model now is characterized by the normal n of this separating hyperplane and an offset, 500–2000 background speakers are used typically.

Z-norm statistics collection: For generative modeling, a set of background speakers can be used in a different way. The likelihoods of background speakers given the target-speaker GMM can be calculated for a set of non-target speakers. The mean and variance of these likelihoods can be stored with the speaker model, and used for score normalization in the test phase. This is known as Z-norming.

- Evaluation Metrics

Applications in speaker detection range from target-sparse applications in intelligence (finding the few utterances from a target speaker in a very large database of recordings) to target-rich applications such as access control (finding the presumably very few break-in attempts in long sequences of genuinely authorized speakers). In a detection trial, the prior probability of a target speaker plays a crucial role. However, these priors cannot be determined by the speaker recognition technology itself, and are given by the application. Therefore, the framework in which a speaker recognition system is evaluated is by a defining a cost function:

$$C_{\text{det}} = C_{\text{miss}} P_{\text{tar}} P_{\text{miss}} + C_{\text{FA}}(1 - P_{\text{tar}}) P_{\text{FA}}$$

Here, the application-specific cost parameters C_{miss} and C_{FA} determine the expected costs made in decision errors. The error rates P_{FA} and P_{miss} indicate the probability of a miss (a not-detected target trial) and false alarm (a falsely detected non-target trials), and must be determined in an evaluation of the system. It can be seen that the target prior P_{tar} governs the cost function.

Acoustic Event Detection

Acoustic event detection (AED) identifies different acoustic events inside and audio stream. The task is inherently harder than speech activity detection because the different event classes can have severely different or similar properties. Often it is hard to model varying durations (even inside the same class of events) and, of course, it is not guaranteed that the sound for a particular event is not a subset of another one (this is a similar problem as in entropy-based compression, see Chapter XXX). AED systems are therefore trained on a case-by-case basis with many hours of data. They are very similar to speaker recognition systems and baseline approaches use Gaussian Mixture Models (GMMs) combined with Hidden Markov Models (HMMs) using a Universal Background Model (UBM). However, recently research has shown that so-called supervector methods can improve event detection. Recent approaches of acoustic event detection therefore compute the mean and standard deviations of the feature trajectories, and use these statistics as

input features for a Support Vectore Machine (SVM). This "GMM-SVM" approach combines the discriminative properties of SVMs with the ability of GMMs to deal with variable length sequences. One can use a linear kernel derived from Kullback-Leiber distance for this:

$$K_{lin}(s_a, s_b) = \sum_{i=1}^{M} (\sqrt{w_i}\Sigma_i^{-\frac{1}{2}}\mu_i^a)(\sqrt{w_i}\Sigma_i^{-\frac{1}{2}}\mu_i^b)^t$$

where s_k is a GMM supervector obtained by pooling together all the Gaussian means μ_{ki} of a means-only MAP-adapted GMM for the sequence k. Σ_i and w_i are the original weight and covariance of each Gaussian on the UBM model used for adaptation. A more detailed description of the approach can be found in the articles referenced.

Speaker Diarization

The goal of speaker diarization is to segment a single-channel audio recording into speaker-homogeneous regions, and cluster these, with the goal of answering the question "who spoke when?" [22]. Figure 9 illustrates the idea. Speaker diarization has a large set of possible and actual applications. Usually, it is used as a front-end (also called upstream) application for different higher level tasks, such as speech recognition, meeting, seminar, or broadcast news navigation, or even dominance detection (based on clues such as who speaks most, who interrupts whom).

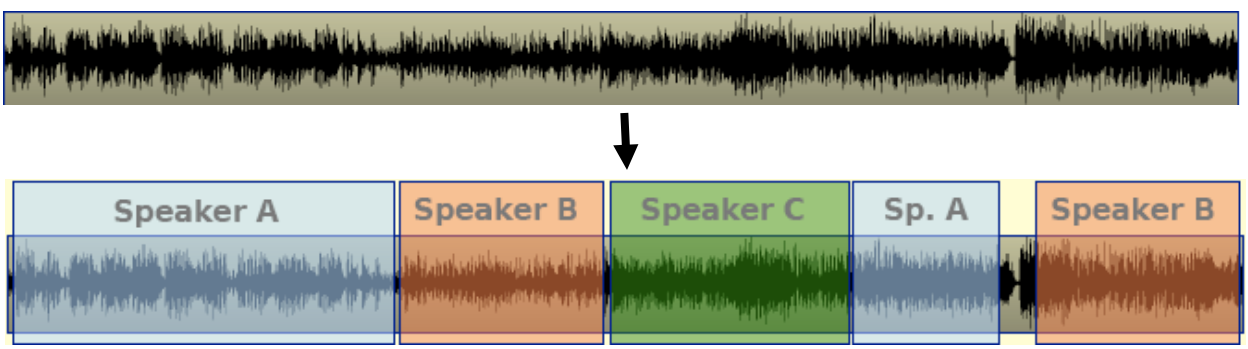


Figure 9. The task of Speaker Diarization is to determine "who spoke when" whithout any prior knowledge about the content of the audio track.

In contrast to speaker recognition or identification, speaker diarization attempts to use no prior knowledge of any kind. This usually means that no specific speaker models are trained for the

speakers that are to be identified in the recording. In practice this means a speaker diarization system has to answer the following questions:

- What are the speech regions?
- How many speakers occur in the recording?
- Which speech regions belong to the same speaker?

Therefore, a speaker diarization system conceptually performs three tasks: First, speech activity detection, second, detect speaker changes to segment the audio data, third, group the segmented regions together into speaker-homogeneous clusters. Some systems unify the two last steps into a single one, i.e., segmentation and clustering is performed in one step. Over the years, many different algorithms have been developed in the speech research community.

Many state-of-the-art speaker diarization systems, use a one-stage approach, i.e., the combination of agglomerative clustering with Bayesian Information Criterion (BIC) (see Chapter XXX) and Gaussian Mixture Models (GMMs, see Chapter XXX) of frame-based cepstral features (MFCCs, see above).

In two-stage speaker diarization approaches, the first step (speaker segmentation) aims at detecting speaker change points and is essentially a two-way classification/decision problem, i.e., for each frame, a decision needs to be made on whether this is a speaker change point or not. After the speaker change detection, the speech segments, each of which contains only one speaker, are then clustered using either top-down or bottom-up clustering. In model-based approaches, pre-trained speech and silence models are used for segmentation. The decision about speaker change is made based on frame assignment, i.e. the detected silence gaps are considered to be the speaker change points. Metric-based approaches are more often used for speaker segmentation. Usually, a metric between probabilistic models of two contiguous speech segments, such as Gaussian Mixture Models, is defined and the decision is made via a simple thresholding procedure. To provide some more technical details about, how a diarization system actually works, we describe one actual system as an example. More details can be found in the original research papers presented under references.

The audio track is usually processed as 19th-order MFCC features using a frame size of 10 ms. A speech activity detector (see above) is used to filter out regions that do not contain speech. The non-speech regions are excluded from the segmentation and clustering. The algorithm is initialized using a much higher amount of clusters than speakers expected in the audio track. Let this

number be k. An initial segmentation is generated by randomly partitioning the audio track into k segments of the same length. Using the initial segmentation, k Gaussian Mixture Models are trained. As classifications based on 10 ms frames are very noisy, a minimum duration of 2.5 seconds is assumed for each speech segment. A majority vote is then used to combine the individual decisions. The algorithm then performs the following loop:

Re-Segmentation: Compute the likelihoods with respect to each Gaussian Mixture Model and vote to determine the assignment of each minimum duration segment to a particular model.

Re-Training: Given the new segmentation of the audio track, train new Gaussian Mixture Models for each of them.

Cluster Merging: Given the new Gaussian Mixture Models, try to find the two models that most likely represent the same speaker. This is done by computing the BIC score (Bayesian Information Criterion) of each of the models and the BIC score of a new GMM trained on the merged segments for two clusters. If the BIC score of the merged Gaussian Mixture Model is smaller than or equal to the sum of the individual BIC scores, the two models are merged and the algorithm loops at the re-segmentation step using the merged Gaussian Mixture Model. If no pair is found, the algorithm stops.

The output of a speaker diarization system consists of metadata describing speech segments in terms of starting time, ending time, and speaker cluster name. This output is usually evaluated against manually annotated ground truth segments. A dynamic programming procedure is used to find the optimal one-to-one mapping between the hypothesis and the ground truth segments so that the total overlap between the reference speaker and the corresponding mapped hypothesized speaker cluster is maximized. The difference is expressed as Diarization Error Rate which is defined by the US National Institute of Standards and Technology (NIST). The Diarization Error Rate (DER) can be decomposed into three additive components: misses (speaker in reference, but not in hypothesis), false alarms (speaker in hypothesis, but not in reference), and speaker-errors (mapped reference is not the same as hypothesized speaker). The difference is expressed as Diarization Error Rate (DER) which is defined as follows:

$$DER = \frac{\sum_{s=1}^{S} \mathrm{dur}(s) \cdot (\max(N_{ref}(s), N_{hyp}(s)) - N_{correct}(s))}{\sum_{s=1}^{S} \mathrm{dur}(s) \cdot N_{ref}}$$

with S being the total number of speaker segments where both reference and hypothesis files contain the same speaker pair(s). It is obtained by comparing the hypothesis and reference speaker turns. The terms $N_{ref}(s)$ and $N_{sys}(s)$ indicate the number of speakers speaking in segment s, and $N_{correct}(s)$ indicates the number of speakers that speak in segment s and have been correctly matched between reference and hypothesis. Segments labelled as non-speech are considered to contain 0 speakers. DER is usually expressed in %. When all speakers and the non-speech in a file are correctly matched the error is 0%.

Speaker Diarization is currently an area of research. Different approaches are investigated, including methods that incorporate spatial information such as video images or the time delay of arrival from different microphones. Research problems include that many speaker diarization systems are not robust enough to be easily ported across different task and data domains. Often parameters of systems are tuned to a particular set of data such as broadcast news or meetings. In a new domain, tuning of parameters often starts from scratch. Also, even variations inside one domain e. g., meeting data recorded at different sites can lead to large variations in performance. Speaker variations caused by emotions or very short interruptions (e. g., shorter than the minimum duration constraint) pose challenges that are yet to be addressed, possibly by multimodal approaches. The greatest challenge is the handling of overlapped speech, which needs to be attributed to multiple speakers.

Other tasks

This chapter only exemplified a couple of acoustic analysis tasks. Many other problems exist, especially in the musical domain, and solutions to them are evolving rapidly as the demand for working searching, organizing, and editing multimedia content creates new challenges to the research community. Chapter XXX will elaborate on Multimedia Information Retrieval, which uses many of the tools presented in the this chapter and in the chapter of visual analysis (see Chapter XXX).

Index Terms

Literature

Porat, Boaz (1997). *A Course in Digital Signal Processing*. New York: John Wiley
Brown, Robert Grover; Hwang, Patrick Y.C. (1996). *Introduction to Random Signals and Applied Kalman Filtering* (3 ed.). New York: John Wiley & Sons
Oppenheim, Alan V.; Schafer, Ronald W.; Buck, John A. (1999). *Discrete-time signal processing*. Upper Saddle River, N.J.: Prentice Hall. pp. 468–471.

Rabiner, Lawrence R., and Gold, Bernard, 1975: Theory and Application of Digital Signal Processing (Englewood Cliffs, New Jersey: Prentice-Hall, Inc.)
Williams, Arthur B & Taylor, Fred J (1995). *Electronic Filter Design Handbook*. McGraw-Hill.
L. R. Rabiner and B. H. Juang, *"Fundamental of Speech Recognition"*, Prentice hall, 1993
B. Gold, N. Morgan, D. Ellis: "*Speech And Audio Signal Processing: Processing And Perception Of Speech And Music*" (Paperback), Wileys and Sons, Second Edition,2011.
Müller, C.(ed.) (2007). "*Speaker Classification I - Fundamentals, Features, and Methods*", Springer, New York - Berlin.

Web Links

Fast Fourier Transforms, online book edited by C. Sidney Burrus, with chapters by C. Sidney Burrus, Ivan Selesnick, Markus Pueschel, Matteo Frigo, and Steven G. Johnson (2008):
http://cnx.org/content/col10550/latest/
FIR Filter FAQ:
http://www.dspguru.com/dsp/faqs/fir

Standards

IEC 268-10 1974 (Peak Programm Meter, Type 1)
ISO Recommendation R. *266*-1975: Preferred frequencies for acoustical measurements

Research Articles

- Fourier, Joseph. (1878). *The Analytical Theory of Heat*. Cambridge University Press (reissued by Cambridge University Press, 2009; ISBN 978-1-108-00178-6)
- Wiener, Norbert (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. New York: Wiley
- James Bergstra, Michael Mandel, and Douglas Eck. Scalable genre and tag prediction with spectral covariance. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 507-512, August 2010.
- K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland, "Overlapped speech detection for improved speaker diarization in multiparty meetings," Proc. ICASSP, pp. 4353–435.
- Chen, S. S. and Gopalakrishnan, P., "Clustering via the bayesian information criterion with applications in speech recognition," Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 2, Seattle, USA, pp. 645-648.
- J. Cohen, T. Kamm, and A. Andreou, "Vocal tract normalization in speech recognition: compensation for system systematic speaker variability," J. Acoust. Soc. Am., vol. 97, no. 5, Pt. 2, pp. 3246–3247, 1995.
- G. Friedland, O. Vinyals, Y. Huang, and C. Mueller, "Prosodic and Other Long-term Features for Speaker Diarization," IEEE Transactions on Audio, Speech, and Language Processing, Vol 17, No 5, pp 985–993, July 2009.

- D.A. Van Leeuwen and M. Konecˇny´, "Progress in the AMIDA Speaker Diarization System for Meeting Data," in Multimodal Technologies for Perception of Humans: International Evaluation Workshops CLEAR 2007 and RT 2007, Baltimore, MD, USA, May 8-11, 2007, Revised Selected Papers. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 475–483.
- J. Ramirez, J. M. Girriz, and J. C. Segura, "Voice activity detection. Fundamentals and speech recognition system robustness," in Robust Speech Recognition and Understanding, M. Grimm and K. Kroschel, Eds., Vienna, Austria, June 2007, p. 460.
- S. Wegmann and L. Gillick, "Why has (reasonably accurate) Automatic Speech Recognition been so hard to achieve?" Tech. Report, Nuance Communications, http://web.mit.edu/kenzie/www/wegmann/wegmann_gillick_why.pdf, 2009.
- S. Tranter and D. Reynolds, "An overview of automatic speaker diarization systems," IEEE TASLP, vol. 14, no. 5, pp. 1557–1565, 2006.

Exercises

1. Implement a simple program that demonstrates the convolution theorem.
2. Smoothing a signal is a low-pass filter. Explain why.
3. Invent some simple mathematical functions for dynamic range compression.
4. Implement a graphic equalizer as described in the chapter. Discuss how wrong application of the tool can distort the audio based on experimentation with your implementation.
5. Imagine applying a graphic equalizer to an image. Explain what would change in the image in the image when you turn the sliders of the lower, mid, and higher frequencies up and down.
6. Older soundcards in the lower price segment often used simple averaging of the sample values for downsampling. Explain what the problem with this approach is.
7. Explain using examples why information cannot be won back by supersampling.
8. How can spectral subtraction work in time space?
9. Implement a Wiener Filter based on the pseudo code presented here. How is the filter applied and what can be improved?
10. Think of a use case for a VU meter and explain how you would want it to work for this use case.
11. Create a program that calculates MFCC features and visualizes them. Input different audio events and notice how the features change.
12. Implement a simple speech activity detection as described in the chapter. Use available corpora from the Internet to train your Gaussians. Measure the classification error with a) different set of parameters when training the Gaussians b) the number of Gaussians c) when testing on the training set d) when testing on a different audio corpus.
13. Discuss possibilities to extend a speech recognition system with video analysis. When do you expect your multimodal system to work well?
14. Explain how you would like to change the behavior of a speaker identification system for these applications: Video retrieval, biometric authorization, a game that gives you a score based on how good you imitate a celebrity's voice.

15. Describe an alternate clustering algorithm for Speaker Diarization that starts with one cluster and splits sub-sequently. Analyze the runtime for the algorithm presented in this chapter and your new one.
16. In the segmentation/clustering algorithm presented in this chapter, the clusters are said to be "purified" in each step by merging two clusters according to the BIC. Provide a colloquial explanation for how this "purification" works. Explain possible problems.
17. Perform the following experiment: Ask a co-student/co-worker to find a video on the Internet in a language that you do not speak and where you do not know the participants. It should contain a conversation of several minutes with at least 4 speakers (a foreign talk show might be a good choice). Do not watch the video, only listen to the audio and perform manual speaker online diarization by saying "speaker 1", "speaker 2". Let your co-worker/co-student rate you: How good are you at assigning the right speakers in a normal and in an overlap situation? How does the situation improve once you look at the video?
18. Pick one of the audio analysis tasks described above. Explain typical expected problems when performing the task as presented here in the following data domains: Recorded voice-over-IP phone conference, a board meeting recorded with a microphone array, a conversation recorded with a cell-phone in a car, a recorded theater performance, broadcast news, an air-traffic control session, a microphone mounted onto a surveillance camera.

Chapter 20: Fundamentals of Visual Processing

Vision is the most powerful human sense. It is commonly believed that a significant part of human perceptual processing, almost two-thirds, is devoted to visual perception. Vision is so important because it provides overview of spatio-temporal information both at overview level as well as local level. Moreover, this is the only sense that can be easily tuned to change its scope to acquire information at different levels of granularity. Because of the generality of vision, humans have a tendency to convert all types of information to visual form for understanding it better. The most interesting example is how text evolved. Text is a visual representation of speech, which is a small fraction of sounds that we understand and recognize. Thus text is an effort to represent sound in visual form. Our language is full of metaphors related to vision; the most common being statements like: Do you *see* my point?

Two common representations of visual information are images (photos) and image sequences (video). A photo is a frozen representation of a visual 'moment' from a specific perspective. A photo provides us spatial layout of optical characteristics resulting from the interaction of optical characteristics of objects and different light sources. A video captures the dynamic happenings from a particular viewpoint. Thus, video is a very rich spatio-temporal representation of optical characteristics due to interactions among objects in an environment.

In many applications of multimedia in early days, video was considered more or less equivalent to multimedia. As the field evolved, it became clear that video is a rich source of information, but unless it is combined with other sources of information, it may be very difficult to recover any information from it. Some important aspects of correlation among different sources and their role in understanding are discussed in a chapter on <Context>.

Visual Information

Many fields in computer sciences address visual processing. We briefly discuss some very closely related fields to provide a general idea of the topics addressed in computer sciences and related areas concerned with some aspects of visual processing of information. The fields most closely related to visual information processing are commonly called computer vision and machine vision. In this section we will refer to them as vision systems or just vision. The main goal of these systems is to extract relevant information from images or video.

Image processing techniques usually transform images into other images; the task of information recovery is left to a human user. Image processing field studies topics such as image enhancement, image compression, and correcting blurred or out-of-focus images. In general, it studies

the process of image formation using different devices and how to improve quality of those images to make them closer to the physical phenomena they represent. In some cases, the goal is to enhance the objects they represent. *Machine vision* algorithms take images as inputs but produce other types of outputs, such as representations for the object contours in an image. Thus, emphasis in machine vision is on recovering information automatically, with minimal interaction with a human. Image processing algorithms are useful in early stages of a machine vision system. They are usually used to enhance particular information and suppress noise.

Computer graphics generates images for a given model using geometric primitives such as lines, circles, and free-form surfaces. Computer graphics techniques play a significant role in visualization and virtual reality. Machine vision is the inverse problem: estimating the geometric primitives and other features from the image. Thus, computer graphics is the synthesis or creation of images, and machine vision is the analysis of images. In the early days of these two fields, there was not much relationship between them, but in the last two decades these two fields have been growing closer. Machine vision is using curve and surface representations and several other techniques from computer graphics, and computer graphics is using many techniques from machine vision to enter models into the computer for creating realistic images. Visualization and virtual reality are bringing these two fields closer.

Pattern recognition classifies numerical and symbolic data using models, generally statistical models, for recognition of objects. Many statistical and syntactical techniques have been developed for classification of patterns for various applications. Techniques from pattern recognition play an important role in machine vision for recognizing objects. In fact, many machine vision techniques rely on *machine learning* algorithms to learn particular object or concept models and then to recognize them. Machine learning techniques are direct derivative of pattern recognition approaches. In pattern recognition the models for objects were assumed to be available, while machine learning assumes that enough data sets are available so that using computational approaches, the system can develop models for the objects to be recognized and then use these models for classifying objects.

Artificial intelligence is concerned with designing systems that are intelligent and with studying computational aspects of intelligence. Artificial intelligence is used to analyze scenes by computing a symbolic representation of the scene contents after the images have been processed to obtain features. Artificial intelligence may be viewed as having three stages: perception, cognition, and action. Perception translates signals from the world into symbols, cognition manipulates symbols, and action translates symbols into signals that effect changes in the world. Many tech-

niques from artificial intelligence may play important roles in all aspects of visual information processing.

Psychophysics, along with cognitive science, has studied human vision for a long time. Many techniques in machine vision are inspired by what is known about human vision. In fact, many researchers in computer vision are more interested in preparing computational models of human vision.

It may be useful to remember the equation

$$\text{Vision} = \text{Geometry} + \text{Measurements} + \text{Interpretation}$$

Thus, vision comprises techniques for estimating features in images, relating feature measurements to the geometry of objects in space, and interpreting this geometric information.

Role of Knowledge

Decision-making usually requires knowledge of the application or goal. Emphasis in vision systems on maximizing automatic operation means that these systems should use knowledge to accomplish this. The knowledge used by the system includes models of features, image formation, models of objects, and relationships among objects. Without explicit use of knowledge, vision systems can be designed to work only in a very constrained environment for very limited applications. To provide more flexibility and robustness, knowledge is represented explicitly and used by the system. In many cases, such knowledge is made available from the context in which these systems work. In Chapter <Context> we will discuss how such models could really be used effectively.

Image Geometry

There are two parts to the image formation process:

- The geometry of image formation, which determines where in the image plane the projection of a point in the scene will be located.
- The physics of light, which determines the brightness of a point in the image plane as a function of scene illumination and surface properties.

Although an understanding of the physics of light is not necessary for understanding the fundamentals of most vision algorithms, such knowledge is useful in building vision systems. Since in our context, we are not concerned with detailed design consideration of vision systems, we will not discuss those processes in details here. The geometry of image formation uses the basic model for the projection of points in the scene onto the image plane as diagrammed in Figure 1.

In this model, the imaging system's center of projection coincides with the origin of the three-dimensional coordinate system. The coordinate system for points in the scene is the three dimensional space spanned by the unit vectors **x**, **y**, and **z** that form the axes.

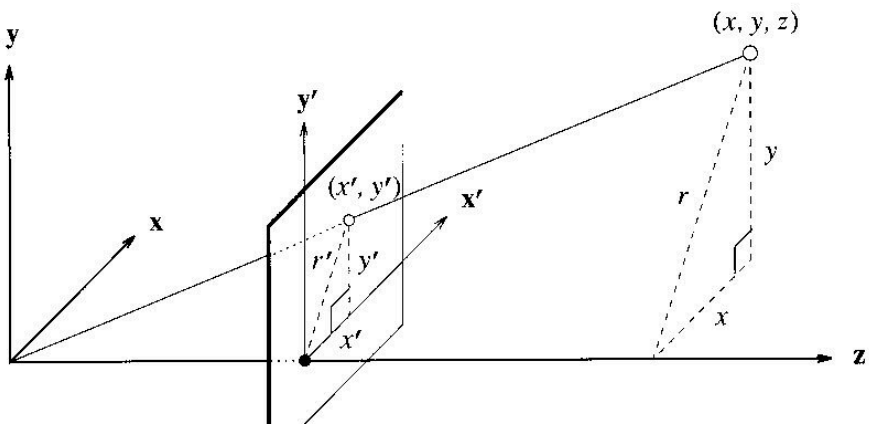


Figure 1 The point on the image plane that corresponds to a particular point in the scene is found by following the line that passes through the scene point and the center of projection of the coordinate system.

A point in the scene has coordinates (x, y, z) and its projection in the camera image appears at location (x', y'). The x coordinate is the horizontal position of the point in space as seen from the camera, the y coordinate is the vertical position of the point in space as seen from the camera, and the z coordinate is the distance from the camera to the point in space along a line parallel to the z axis. The *line of sight* of a point in the scene is the line that passes through the point of interest and the center of projection. The line drawn in Figure 1 is a line of sight.

The image plane is parallel to the x and y axes of the coordinate system at a distance / from the center of projection, as shown in Figure 1. This figure helps us understand how a camera converts a three-dimensional world to a two-dimensional image by essentially a many-to-one mapping. This means that without the knowledge about the domain and the context, an image cannot be unambiguously interpreted. This is the fundamental reason behind the complexity in visual processing.

Sampling and Quantization

A continuous function must be converted to a discrete form for representation and processing using a digital computer. The interface between the optical system that projects a scene onto the

image plane and the computer must sample the image at a finite number of points and represent each sample within the finite word size of the computer. This requires applying sampling and quantization steps. Each image sample is called a pixel. We will assume that images are sampled on a regular grid of squares so that the horizontal and vertical distances between pixels are the same throughout the image. Many cameras acquire an analog image, which is then sampled and quantized to convert it to a digital image. The sampling rate determines how many pixels the digital image will have (the image resolution), and quantization determines how many intensity levels will be used to represent the intensity value at each sample point. The image processing books (see further reading section) discuss the factors that should be considered in selecting appropriate sampling and quantization rates to retain the important information in images. In most modern cameras, the sampling and quantizing rates are predetermined and specified as number of pixels (as MP, or million pixels) available on the chip used in the camera. In this case, the images are directly acquired in digital form.

Levels of Computation

An image usually contains several objects. A vision application usually involves computing certain properties of an object, not the image as a whole. To compute properties of an object, individual objects must first be identified as separate objects; then object properties can be computed by applying calculations to the separate objects. For considering computational aspects of vision algorithms, it helps to consider each algorithm in terms of its input-output characteristics. By considering nature and the level of these operations it is possible to consider how best to implement these operations. Note that the input to a computer vision system is an image, and the output, unlike that of image processing systems, is some symbolic quantity denoting identity or location of an object, for instance. The amount of data processed by a vision system is very large, and that makes the computational requirements very demanding. Since we want to discuss characteristics of operations to predict their computational requirements, we classify the levels of operations and study their general characteristics.

Point Level

Some operations produce an output based on only a point in an image. Thresholding is an example of a point operation. A thresholding algorithm produces output values that depend only on the input value, for a preset threshold. Thus, if I(x,y), and O(x,y) are input and output images, respectively, then for a point operation,

$$O(x,y) = f\,(I\,(x+k,y+l))$$

Meaning that in the output image, the attribute or intensity value at a point depends only on the attribute value at a corresponding point.

This operation can be efficiently implemented using a lookup table.

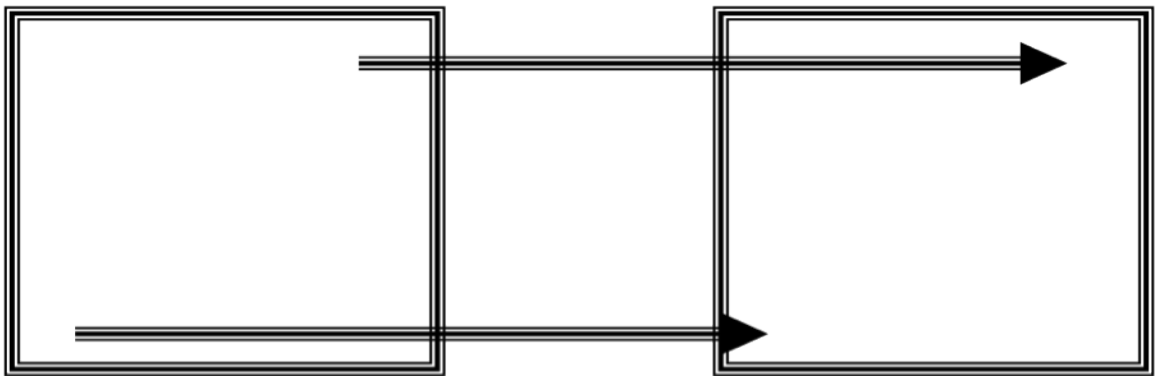


Figure 2. *Top:* Point operations are applied to individual image pixels and produce an output image as the result. *Bottom left:*

Local Level

A local operation produces an output image in which the intensity at a point depends on the neighborhood of the corresponding point in the input image.

Thus, for a local operation,

$$O(x,y) = f\,(I\,[Nbr(x,y)])$$

Where Nbr[x,y] denotes a neighborhood of point (x,y). Most commonly used neighborhoods are 3 X 3 windows centered around the point. It is possible to use larger size windows depending on the operation under consideration.

An example of such an operation is shown in Figure 3. Smoothing and edge detection are local operations. Since these operations require values from a neighborhood in the input image, array processors or Single Instruction, Multiple Data (SIMD) machines may be suitable for implementing these operations. In general, these operations can be easily implemented on parallel machines and can often be performed in real time.

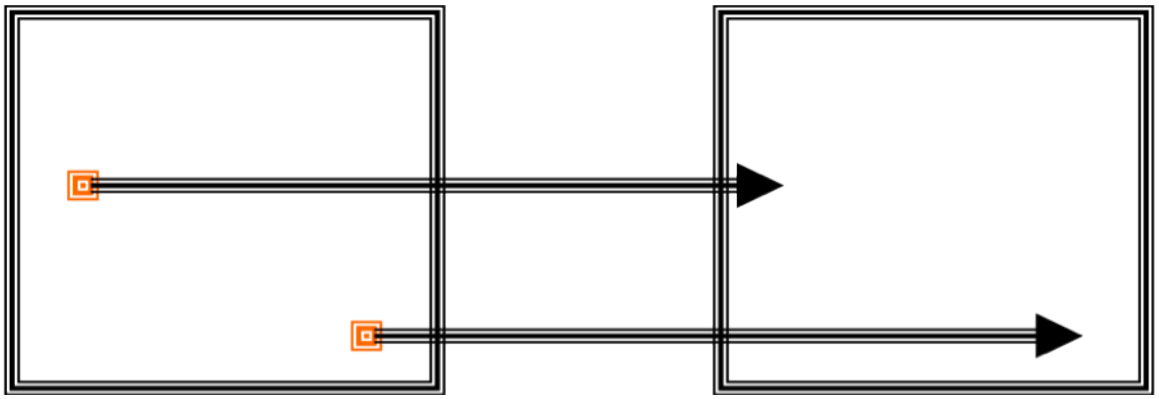


Figure 3. *Top:* Local operations are applied to pixel neighborhoods and produce an output image as the result.

Global Level

The output of certain operators depends on the whole picture. Such operations are called global operations:

$$P = f(I(x,y))$$

This operation is shown in Figure 4. The output of these operators may be an image or it may be symbolic output. A histogram of intensity values and the Fourier transform are global operations. We will see that most operations at higher levels are global in nature and are most expensive in terms of time requirements.

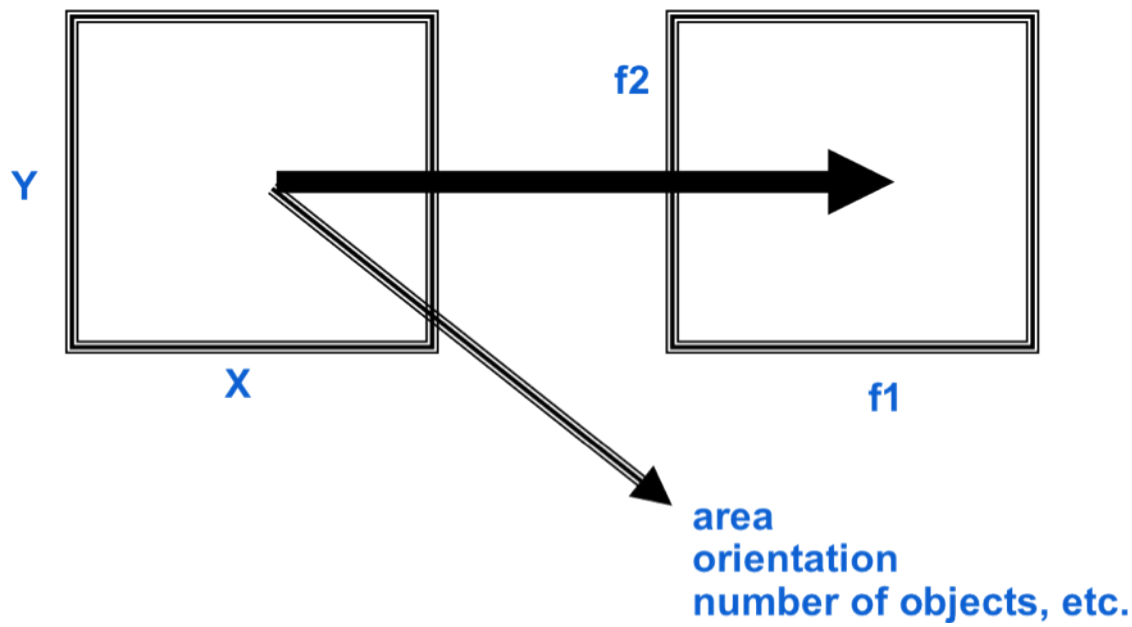


Figure 4. An example of a global operation: an image (left) and its histogram (right). A histogram is a plot of the number of pixels at each gray value contained in the image.

Object Level

Most applications of computer vision require properties to be computed at the object level. Size, average intensity, shape, and other characteristics of an object must be computed for the system to recognize it. Many other characteristics of an object must be determined for detecting features for recognition of an object. This leads to an interesting, but very difficult questions: What is an object? How do we find objects? In Figure 5, how many objects do we see? Some people say 5, some 6 and some may even say 7. All of them are right. We will see that an object is defined in a particular context. In fact, many operations in vision are performed to find where a particular object is located in an image. Objects in images pose a *catch-22* situation. We must use all points that belong to an object to compute some of its characteristics, but we must use those characteristics to identify those points. We will see that significant efforts are spent to solve the *figure-ground* problem (separation of foreground pixels from background pixels) to group points into objects. To understand the contents of an image, a vision system must perform several operations at the object level.

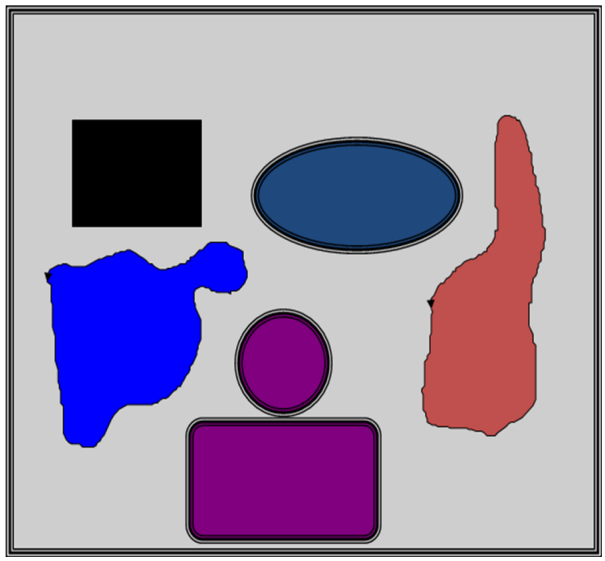


Figure 5. How many objects do we see in this figure? Depending on your definition of an object you may see 5, 6, or 7 objects in this figure.

Thresholding

One of the most important problems in a vision system is to identify the sub-images that represent objects. This operation, which is so natural and so easy for people, is surprisingly difficult for computers. Let us consider that we want to mark all points in an image that belong to an object of interest, commonly called foreground, as 1 and all other points as 0. Thus we convert an image to a binary image that gives us a mask for all object points. Such a binary image is obtained using an appropriate segmentation of a gray scale image. If the intensity values of an object are in an interval and the intensity values of the background pixels are outside this interval, a binary image can be obtained using a thresholding operation that sets the points in that interval to 1 and points outside that range to 0. Thresholding is a method to convert a gray scale image into a binary image so that objects of interest are separated from the background. For thresholding to be effective in *object-background separation,* it is necessary that the objects and background have sufficient contrast and that we know the intensity levels of either the objects or the background. In a fixed thresholding scheme, these intensity characteristics determine the value of the threshold.

Let us assume that a binary image $B(i,j)$ is the same as a thresholded gray image $F_{Thr}[i, j]$ which is obtained using a threshold T for the original gray image $F[i, j]$. Thus,

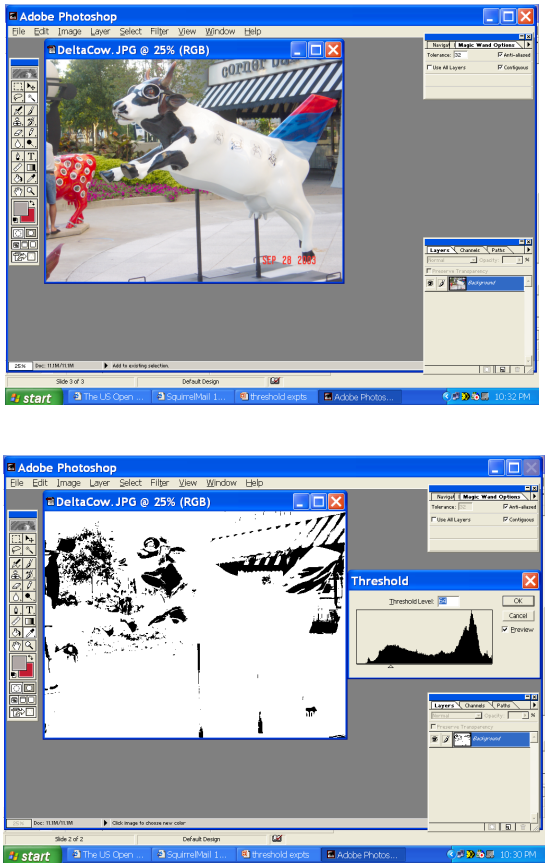
$$B[i, j] = 1 \text{ if } F[i, j] > T$$

$$= 0 \text{ otherwise.}$$

If it is known that the object intensity values are in a range [T1, T2], then we may use

$$B[i,j] = 1 \text{ if } T1 < F[i, j] < T2$$
$$= 0 \text{ otherwise.}$$

The results of producing an image using different thresholds are shown in Figure 6. Note how knowledge about the application domain is required in selecting the threshold. The same threshold values may not work in a new set of images acquired under different conditions. The threshold is usually selected on the basis of experience with the application domain. In some cases, the first few runs of the system may be used for interactively analyzing a scene and determining an appropriate value for the threshold. Automatic thresholding of images is often the first step in the analysis of images in machine vision systems. Many techniques have been developed for utilizing the intensity distribution in an image and the knowledge about the objects of interest for selecting a proper threshold value automatically.



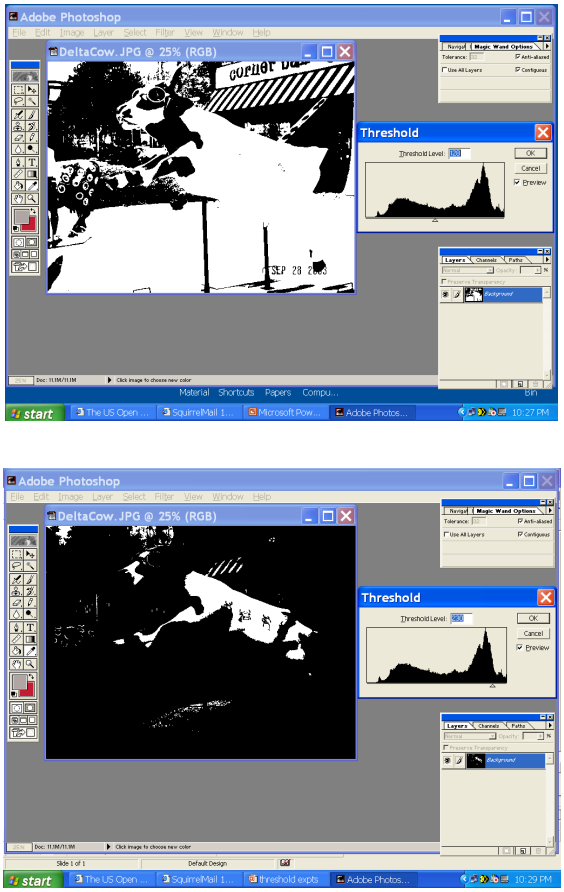


Figure 6. An image and several binary images obtained at different threshold values.

Geometric Properties

Suppose that a thresholding scheme has given us objects in an image. The next step is to recognize and locate objects. In most industrial applications, the camera location and the environment are known. Using simple geometry, one may find the three-dimensional locations of objects from their two dimensional positions in images. Moreover, in most applications the number of different objects is not large. If the objects are different in size and shape, the size and shape features of objects may be determined from their images to help the system recognize them. Many applications utilize some simple features of regions for determining the locations of objects and for recognizing them (e.g., size, position, orientation). If there are several objects, one can compute these features for each object. A connected component or a region usually represents an object. The concept of connectedness and the algorithms for finding connected components in an image will be discussed later in this chapter.

Size

In general, for a binary image it is well known that the area A is given by the number of pixels in the image of the object.

Position

The position of an object in an image plays an important role in many applications. There are different ways to specify the position of an object, such as using its enclosing rectangle or centroid. In industrial applications, objects usually appear on a known surface, such as a table, and the position of the camera is known with respect to the table. In such cases, an object's position in the image determines its spatial location. The position of an object in an image may be defined using the center of area of the object image. Though other methods such as the enclosing rectangle of the object image may be used, the center of area is a point and is relatively insensitive to noise in the image.

Binary Algorithms

Segmenting object pixels from background pixels is a difficult problem. We will not address this problem here. Let us assume here that somehow an object can be defined and, using a predicate, the points of an image belonging-to an object may be labeled. The problem then is to group together all points of an image that are labeled as object points into an object image. In this chapter we will assume that all such points are spatially close. This notion of *spatial proximity* requires a more precise definition so that an algorithm may be devised to group spatially close points into a component. For this purpose, let us introduce some definitions.

Neighbors

A pixel in a digital image is spatially close to several other pixels. In a digital image represented on a square grid, a pixel has a common boundary with four pixels and shares a corner with four additional pixels. We say that two pixels are 4 -neighbors if they share a common boundary. Similarly, two pixels are 8-neighbors if they share at least one corner. For example, the pixel at location [i,j] has 4-neighbors [i + 1,j], [i -1,j], [i,j + 1], and [i,j -1]. The 8-neighbors of the pixel include the 4-neighbors plus [i + 1, j + 1], [i + 1, j - 1], [i-1, j + 1] and [i -1, j -1]. A pixel is said to be 4-connected to its 4-neighbors and 8-connected to its 8-neighbors (see Figure 7).

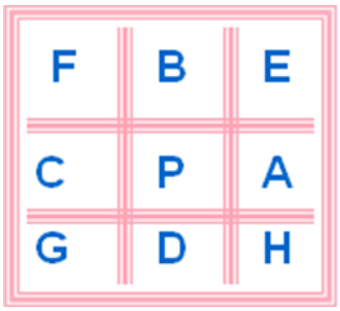


Figure 7. The 4- and 8-neighborhoods for a rectangular image tessellation. Pixel [i, j] is located in the center of each figure. For point P, its 4-neighbors are A, B, C, and D, and it's 8-neighbors will additionally include E, F, G, and H.

Path

A path from the pixel at $[i_0, j_0]$ to the pixel at $[i_n, j_n]$ is a sequence of pixel indices $[i_0, j_0]$, such that the pixel at $[i_k, i_k]$ is a neighbor of the pixel at $[i_{k+1}, i_{k+1}]$ for all k with 0 ~ k ~ n - 1. If the neighbor relation uses 4-connection, then the path is a 4-path; for 8-connection, the path is an 8-path. Simple examples of these are shown in Figure 7. The set of all 1 pixels in an image is called the *foreground* and is denoted by S.

Connectivity

A pixel *p* in S is said to be *connected* to *q* in S if there is a path from *p* to *q* consisting entirely of pixels of S. Note that connectivity is an equivalence relation. For any three pixels *p, q,* and *r* in S, we have the following properties:

1. Pixel *p* is connected to *p* (reflexivity).
2. If *p* is connected to *q,* then *q* is connected to *p* (commutativity).
3. If *p* is connected to *q* and *q* is connected to *r,* then *p* is connected to *r* (transitivity).

Connected Components

A set of pixels in which each pixel is connected to all other pixels is called a *connected component*.

Background

The set of all connected components of *S* (the complement of S) that have points on the border of an image is called the *background*. All other components of *S* are called *holes*.

Let us consider the simple picture shown in Figure 8. How many objects and how many holes are in this figure? If we consider 4-connectedness for both foreground and background, there are four objects that are 1 pixel in size and there is one hole. If we use 8-connectedness, then there is one object and no hole. Intuitively, in both cases we have an ambiguous situation. To avoid this and similar awkward situations, different connectedness should be used for objects and backgrounds. If we use 8-connectedness for *S*, then 4- connectedness should be used for *S*.

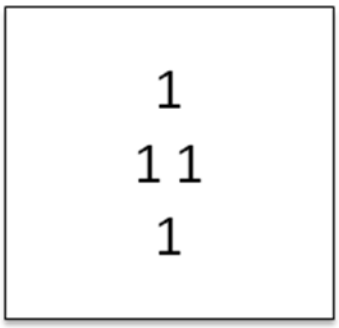


Figure 8. Considering background and foregrounds as different connected component helps.

One of the most common operations in vision systems is finding the connected components in an image. The points in a connected component form a candidate region for representing an object. As mentioned earlier, in computer vision most objects have surfaces. Points belonging to a surface project to spatially close points. The notion of "spatially close" is captured by connected components in digital images.

A component labeling algorithm finds all connected components in an image and assigns a unique label to all points in the same component. Figure 9 shows an image and its labeled connected components. In many applications, it is desirable to compute characteristics (such as size, position, orientation, and bounding rectangle) of the components while labeling these components.

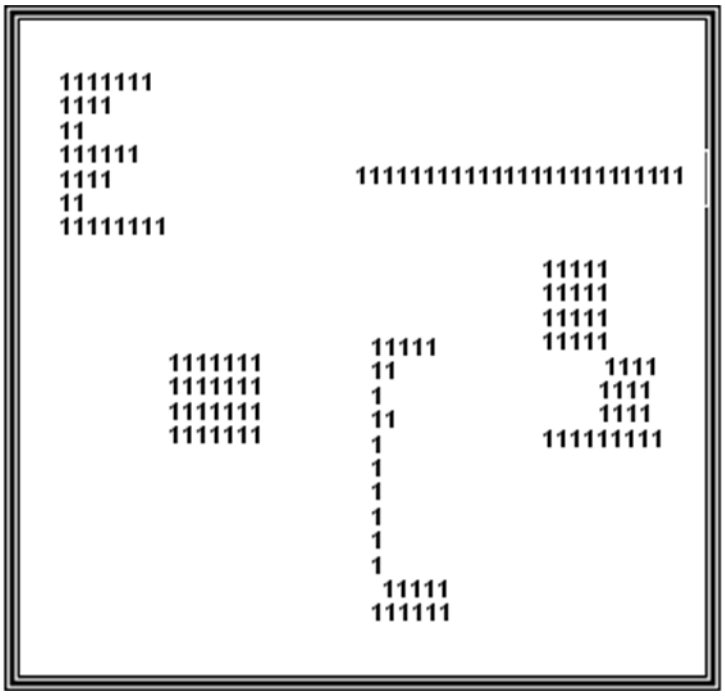


Figure 9. A binary image with 7 4-connected and 5 8-connected regions (connected components).

Basic Visual Processing Operations: Finding Edges and Regions

Understanding an image requires identifying different objects in it and knowing spatial relationships among them. The understanding of an image is to partition it into clearly marked regions of pixels corresponding to different objects. Such partitions are obtained from the characteristics of the intensity values of the pixels in the image. There are two approaches to partitioning an image into regions: region-based segmentation and boundary estimation using edge detection. Ideally, a region should be bounded by a closed contour, but in many cases using current techniques this is not the case. In principle, region segmentation and edge detection should yield complementary results. Unfortunately, in real images due to noise and other factors, neither region segmentation nor edge detection provides perfect information.

In the next section, we describe basic concepts and techniques for region and edge detection. This is an active area of research. Our goal here is to provide understanding of basic concepts related to region finding and edge detection.

Regions

Pixels must be assigned to regions using criteria that distinguishes them from the rest of the image and helps in separating objects from each other and from the background. Two very important principles in segmentation are *similarity* and *spatial proximity*. Two pixels may be assigned to the same region if they have similar characteristics and if they are close to one another. For example, a specific measure of similarity between two pixels is the difference between the intensity values, and a specific measure of spatial proximity is Euclidean distance. The principles of similarity and proximity are derived from the knowledge that points on the same object will usually project to pixels in the image that are spatially close and have similar intensity values. We can group pixels in an image using these simple assumptions and then use domain-dependent knowledge to match regions to object models. In simple situations, segmentation can be done with thresholding and component labeling, complex images may require more sophisticated techniques to assign pixels to regions that correspond to parts of objects.

Region Segmentation

Let's discuss the process of region formation, or segmentation, more precisely. Given a set of image pixels I and a homogeneity predicate $P(.)$, find a partition S of the image I into a set of n regions,

$$\bigcup_{i=1}^{n} R_i = I.$$

The homogeneity predicate and partitioning of the image have the properties that any region satisfies the predicate

$P(\sim)$ = True for all R_i, and any two adjacent regions cannot be merged into a single region that satisfies the predicate

$P(\sim \bigcup R_j)$ = False.

The homogeneity predicate $P(.)$ defines the conformity of all points in the region to the region model.

The process of converting an image into a binary image is a simple form of segmentation where the image is partitioned into two sets corresponding to objects and background. The algorithms for thresholding to obtain binary images can be generalized to more than two levels. To make segmentation robust to variations in the scene, the algorithm should be able to select an appropriate threshold automatically by analyzing image intensity values in the image. The knowledge about the intensity values of objects should not be hard-wired into an algorithm; the algorithm should use knowledge about the relative characteristics of intensity values to select the appropriate threshold. This simple idea is useful in many computer vision algorithms.

Edges

An edge point represents a point that separates two regions corresponding to two different objects. Thus, an edge in an image is a significant local change in the image intensity. Algorithmically, this is usually associated with a discontinuity in either the image intensity or the first derivative of the image intensity. Discontinuities in the image intensity can be either (1) *step* discontinuities, where the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side, or (2) *line* discontinuities, where the image intensity abruptly changes value but then returns to the starting value within some short distance. However, step and line edges are rare in real images. Because of low-frequency components or the smoothing introduced by most sensing devices, sharp discontinuities rarely exist in real signals. Step edges become *ramp* edges and line edges become *roof* edges, where intensity changes are not instantaneous but occur over a finite distance. Most edges in images are combination of step and line discontinuities. In computer vision, many different approaches have been developed to deal with complex images. Here we will discuss concepts related to edge detection using a general step model.

It is important to define some terms before we discuss edge detection operators to understand precisely what they do and how their results should be analyzed.

An *edge point* is a point in an image with coordinates [i,j] at the location of a significant local intensity change in the image. An *edge fragment* corresponds to the i and j coordinates of an edge and the *edge orientation* e, which may be the gradient angle. An *edge detector* is an algorithm that produces a set of edges {edge points or edge fragments} from an image. A *contour* is a list of edges or the mathematical curve that models the list of edges. *Edge linking* is the process of forming an ordered list of edges from an unordered list. By convention, edges are ordered by traversal in a clockwise direction. Edge following is the process of searching the filtered image to determine contours.

Edge detection is essentially the operation of detecting significant local changes in an image. In one dimension, a step edge is associated with a local peak in the first derivative. The gradient is a measure of change in a function, and an image can be considered to be an array of samples of some continuous function of image intensity. By analogy, significant changes in the gray values in an image can be detected by using a discrete approximation to the gradient. The gradient is the two-dimensional equivalent of the first derivative and is defined as a *vector.* It is common practice to approximate the gradient magnitude by absolute values. As a very simple approximation one may consider edgeness in directions x and y as

$$E_x(x, y) = F(x + 1, y) - F(x,y)$$
$$E_y(x,y) = F(x, y+1) - F(x, y)$$

Then,the greatest magnitude is in direction

$$\alpha = \tan^{-1} (E_y / E_x)$$

and the edge magnitude is

$$E_m = (Ex^2 + Ey^2)^{1/2}$$

where the angle α is measured with respect to the *x* axis.

Note that the magnitude of the gradient is actually independent of the direction of the edge. Such operators are called *isotropic operators.*

Algorithms for edge detection contain three steps:

Filtering: Since gradient computation based on intensity values of only two points are susceptible to noise and other vagaries in discrete computations, filtering is commonly used to improve the performance of an edge detector with respect to noise. However, there is a trade-off between edge strength and noise reduction.

Enhancement: In order to facilitate the detection of edges, it is essential to determine changes in intensity in the neighborhood of a point. Enhancement emphasizes pixels where there is a significant change in local intensity values and is usually performed by computing the gradient magnitude.

Detection: We only want points with strong edge content. However, many points in an image have a nonzero value for the gradient, and not all of these points are edges for a particular application. Therefore, some method should be used to determine which points are edge points. Frequently, thresholding provides the criterion used for detection. Figure 10 shows an image and strong edges detected in this image.

Segmentation Using Split and Merge

A simple intensity-based segmentation usually results in too many regions. Even in images where most humans see very clear regions with constant intensity value, the output of a thresholding algorithm may contain many extra regions. The main reasons for this problem are high-frequency noise and a gradual transition between intensity values in different regions. After the initial intensity-based region segmentation, the regions may need to be refined or reformed. Several approaches have been proposed for postprocessing such regions obtained from a simple segmentation approach. Some of these approaches use domain-dependent knowledge, while other approaches use knowledge about the imaging process. The refinement may be done interactively by a person or automatically by a computer. In an automatic system, the segmentation will have to be refined based on object characteristics and general knowledge about the images. Automatic refinement is done using a combination of split and merge operations. Split and merge operations eliminate false boundaries and spurious regions by merging adjacent regions that belong to the same object, and they add missing boundaries by splitting regions that contain parts of different objects. Some possible approaches for refinement include:

Figure 10. An image and edges detected in this image.

- Merge adjacent regions with similar characteristics.
- Remove questionable edges.
- Use topological properties of the regions.
- Use shape information about objects in the scene.
- Use semantic information about the scene.

The first three approaches use only information about image intensity combined with other domain-independent characteristics of regions. The other two use domain dependent knowledge. The literature about the segmentation of Images that usually combine above techniques is vast and ever-growing. Such techniques usually result in segmentation shown in Figure 11. We will point to some appropriate sources in further readings.

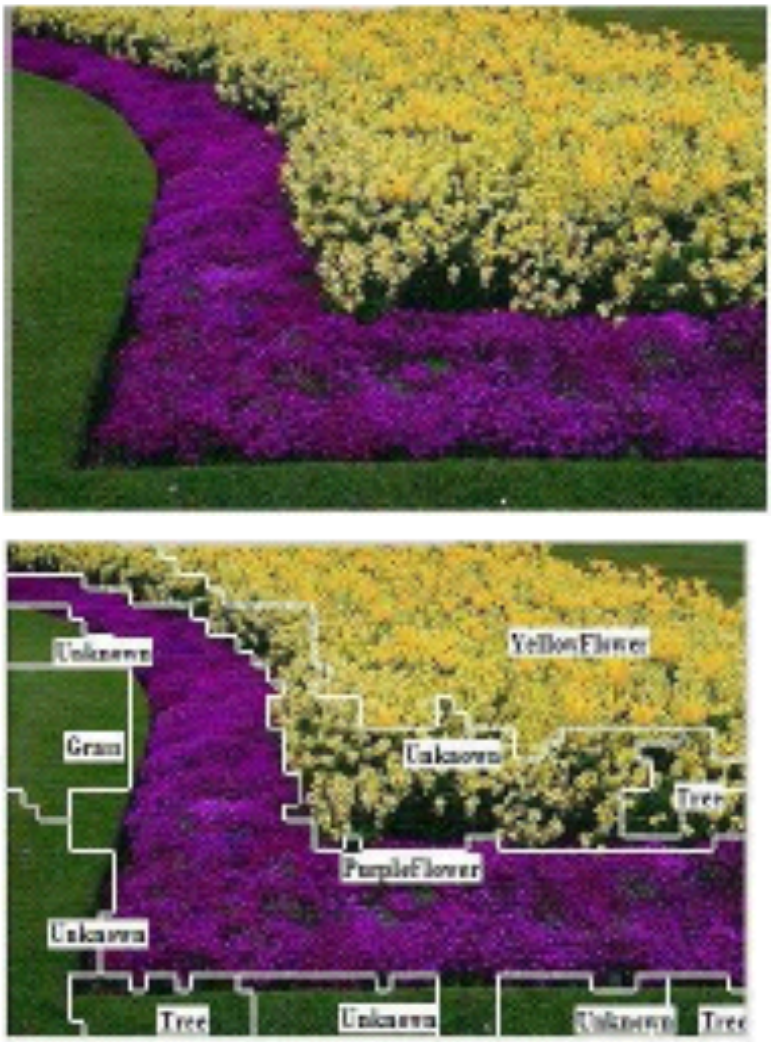


Figure 11. A natural scene image and its segmentation into many known components.

Salient Visual Attributes

Humans use some common visual attributes to describe and understand objects and concepts in images. These attributes have been widely studied by psychologists to understand how human visual processing works. These techniques have been widely studied and applied in image understanding and image retrieval systems. In this section, we study three prominent characteristics commonly used in different applications.

Color

An imaging system may use multiple images, commonly called channels, one for each sensor responding to special frequency of light in the imaging system. In color imagery, there are usually three sensors with spectral responses that cover the red, green, and blue regions of the visible spectrum (RGB space). RGB are called primary colors because human visual system is sensitive to these colors and all other colors perceived by humans are basically a combination of these colors. It is common to work with normalized RGB values, so the set of all possible colors are in a unit cube with corners at (0,0,0) and (1,1,1) in the RGB space. Although RGB are the primary colors, in applications many different transformations are used to suit characteristics of devices and computational requirements.

Hue, brightness, and saturation are considered important characteristics of color based on human perception. Hue is determined by the dominant wavelength in the spectral distribution of light wavelengths. Hue refers to the color names we commonly use. The brightness is a measure of the overall amount of light contained in all three color components. We can think of the brightness as the magnitude of the light as perceived. One measures value or energy at a particular point as brightness or darkness of the pixel. The saturation refers to the dominance of hue in the color. In a way it refers to whether a particular color dominates or the pixel is balanced combination of colors. The pixels that are closer to a specific color, say red, have higher saturation value. In a color wheel commonly used to select colors in many applications, the angle and saturation are selected on the color wheel as shown in the Figure 12 and the intensity or value is selected using a slider. This figure shows how hue and saturation are related to the colors we perceive.

The HSI (or HSV) color model represents a color in terms of hue, saturation, and intensity. This model has been very popular in many applications of image processing. However, there are many other models that are the result of similar transformation of colors by combining the primary colors in different ways that have been defined for specific applications. The RGB components of an image can be converted to the HSI color representation. There are many converters available on line. Most of the color pickers used in word processing and presentation systems use HSV representation rather than RGB representation in their selection palette because of its easy understanding by human beings.

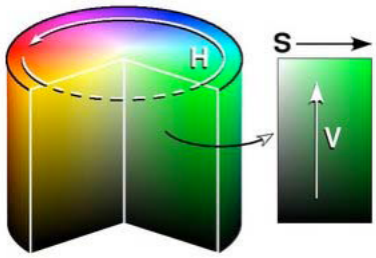


Figure 12. HSV color space. (From: http://coecsl.ece.illinois.edu/ge423/spring05/group8/FinalProject/HSV_writeup.pdf

May be replaced in the final version.)

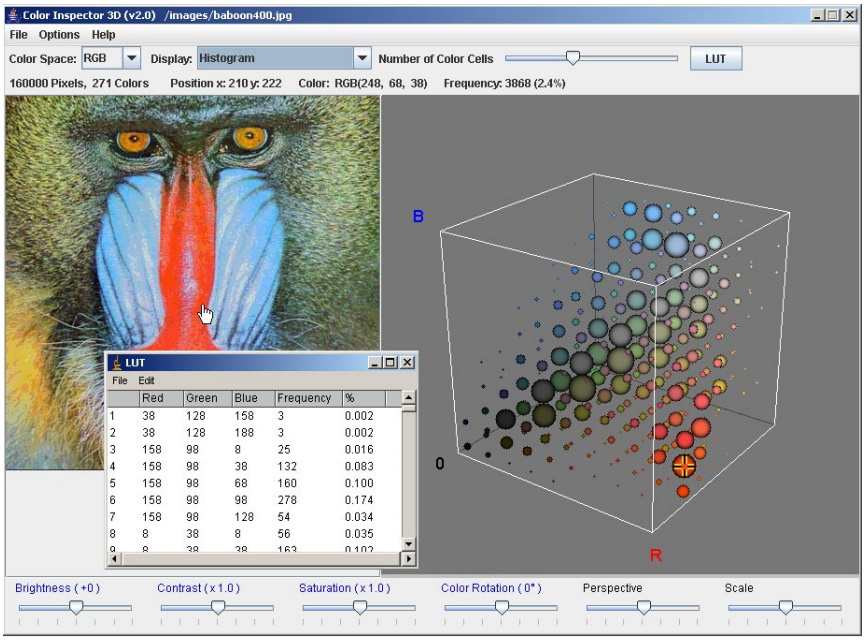


Figure 13. Forming Color Histograms. **(Figure may be replaced in the final version.)**

Color Histograms

In many applications, color histograms are used to represent either an object or an image. The color histogram is used to represent variations in color in the region. We discussed intensity value color histograms earlier in this chapter, where we collected number of pixels with a particular value in the image. Since a color pixel has three values corresponding to the three color components, the histogram may be a three-dimensional histogram. If each color is represented using a 256 values then one needs a histogram with 256 X 256 X 256 bins. To simplify computations, one may consider the intensity ranges for each component to be represented as 4 bins such that bins 0, 1, 2, and 3 contain ranges of values corresponding to (0 to 63), (64 to 127), (128 to 191), and (192 to 255), respectively. This will reduce the total color space to 64 bins (4 X 4 X4). One can prepare and represent an image using such a histogram. These steps are shown in Figure 13.

The main use of color histograms is to compare one image with other image based on the color distribution in these images. Suppose one is given two images I_1 and I_2 and their histograms are H_1 and H_2. To compare these images one can use a distance measure between the two histograms. Two popular distance measures are the well known L_1 and L_2 distance measures. These result in the distance between two images based on their histograms as,

L_1 distance:

$$\mathbf{D_{H12} = \Sigma_i \ | H_1(i) - H_2(i) |}$$

and L_2 distance:

$$\mathbf{D_{HK} = \Sigma_i \ (H_1(i) - H_2(i))^2}$$

One may also consider quadratic distance between two histograms. Consider that each histogram is a vector and consider that there is a matrix C (N X N) that gives similarity of each color i with color j. Then we define quadratic distance between two histograms as:

$$\mathbf{D_Q (H_1,H_2) = (H_1 - H_2)^t \ \ C (H_1 - H_2)}$$

As is clear, this formulation allows us to consider similarity of individual colors and hence some subjective elements may be introduced easily in this formulation.

Texture

Texture is characterized by the spatial distribution of gray levels in a neighborhood. Thus, texture cannot be defined for a point. The resolution at which an image is observed determines the scale at which the texture is perceived. For example, in observing an image of a tiled floor from a large distance we observe the texture formed by the placement of tiles, but the patterns within the tiles are not perceived. When the same scene is observed from a closer distance, so that only a few tiles are within the field of view, we begin to perceive the texture formed by the placement of detailed patterns composing each tile. For our purposes, we can define texture as repeating patterns of local variations in image intensity, which are too fine to be distinguished as separate objects at the observed resolution. Thus, a connected set of pixels satisfying a given gray-level properties which occur repeatedly in an image region constitutes a textured region. A simple example is a repeated pattern of dots on a white background. Some prominent textures are shown in Figure 14.

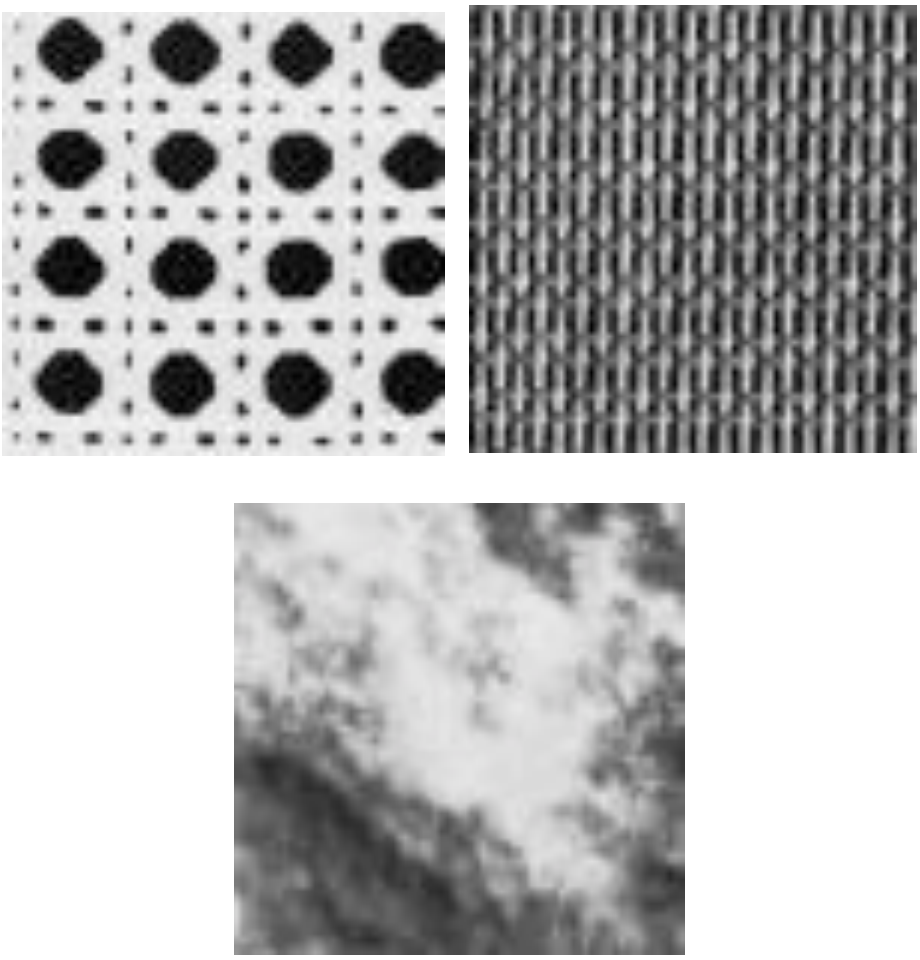


Figure 14. Prominent textures. The first two show regular structure, while the third one shows a texture that is characterized by its statistical characteristics.

One common approach to measure and characterize texture is commonly called as Tamura textures. This approach defines six textural features: coarseness, contrast, directionality, line-likeness, regularity and roughness. These features correspond well to what humans usually use to characterize textures and have become popular.

Coarseness has a direct relationship to scale and repetition rates and is considered as the most fundamental texture feature. An image contains textures at several scales; coarseness aims to identify the largest size at which a texture exists, even where a smaller micro texture exists.

Contrast aims to capture the dynamic range of grey levels in an image, together with the polarisation of the distribution of black and white.

Directionality is a global property over a region. The feature described does not aim to differentiate between different orientations or patterns, but measures the total degree of directionality.

Tamura Image is a notion where we calculate a value for the three features at each pixel and treat these as a spatial joint coarseness-contrast-directionality (CND) distribution, in the same way as images can be viewed as spatial joint RGB distributions. We extract color histogram style features from the Tamura CND image, both marginal and 3D histograms. The regional nature of texture meant that the values at each pixel were computed over a window.

Video processing: Analyzing Dynamic Scenes

The input to a dynamic scene analysis system is a sequence of image frames taken from a changing world. The camera used to acquire the image sequence may also be in motion. Each frame represents an image of the scene at a particular instant in time. The changes in a scene may be due to the motion of the camera, the motion of objects, illumination changes, or changes in the structure, size, or shape of an object. It is usually assumed that the changes in a scene are due to camera and/or object motion, and that the objects are either rigid or quasi-rigid. The system must detect changes, determine the motion characteristics of the observer and the objects, characterize the motion using high-level abstraction, recover the structure of the objects, and recognize moving objects. **In** applications such as video editing and video databases, it may be required to detect *macro* changes in a sequence. These changes will partition the segment into many related segments exhibiting similar camera motion or a similar scene in a sequence.

A scene usually contains several objects. An image of the scene at a given time represents a projection of the scene, which depends on the position of the camera. There are four possibilities for the dynamic nature of the camera and world setup:

1. Stationary camera, stationary objects (SCSO)
2. Stationary camera, moving objects (SCMO)
3. Moving camera, stationary objects (MCSO)
4. Moving camera, moving objects (MCMO)

For analyzing image sequences, different techniques are required in each of the above cases. The first case is simply static-scene analysis. Many applications require information extracted from a dynamic environment; in some cases a vision system must understand a dynamic process from a single viewpoint. In applications such as mobile robots or autonomous vehicles, a vision system must analyze an image sequence acquired while in motion. Recovery of information from a mobile camera requires different techniques than those when the camera remains stationary. A sequence of image frames offers much more information to aid in understanding a scene but sig-

nificantly increases the amount of data to be processed by the system. However, research in dynamic-scene analysis has shown that the recovery of information in many cases is easier in dynamic scenes than in static scenes. In dynamic-scene analysis, SCMO scenes have received the most attention. In analyzing such scenes, the goal is usually to detect motion, to extract masks of moving objects for recognizing them, and to compute their motion characteristics. MCSO and MCMO scenes are very important in navigation applications. MCMO is the most general and possibly the most difficult situation in dynamic scene analysis, but it is also the least developed area of computer vision.

Change Detection

Detection of changes in two successive frames of a sequence is a very important first step for many applications. Any perceptible motion in a scene results in some change in the sequence of frames of the scene. Motion characteristics can be analyzed if such changes are detected. A good quantitative estimate of the motion components of an object may be obtained if the motion is restricted to a plane that is parallel to the image plane; for three-dimensional motion, only qualitative estimates are possible. Any illumination change in a scene will also result in changes in intensity values, as will scene changes in a TV broadcast or a movie. Most techniques for dynamic-scene analysis are based on the detection of changes in a frame sequence. Starting with frame-to-frame changes, a global analysis of the sequence may be performed. Changes can be detected at different levels: pixel, edge, or region. Changes detected at the pixel level can be aggregated to obtain useful information with which the computational requirements of later phases can be constrained. We discuss different techniques for change detection. We will discuss one of the simplest, yet one of the most useful change detection techniques, *difference pictures*. In a special case, this technique is called background subtraction, when one of the frame is known to be the background because in that case the resulting difference pictures show areas corresponding to moving objects.

Difference Pictures

The most obvious method of detecting change between two frames is to directly compare the corresponding pixels of the two frames to determine whether they are the same. In the simplest form, a binary difference picture $DPjk(x, y)$ between frames $F(x, y, j)$ and $F(x, y, k)$ is obtained by:

$$DPjk(x, y) = 1 \text{ if } |F(x,y,j) - F(x,y,k)| > T$$
$$= 0 \text{ otherwise}$$

where T is a threshold.

In a difference picture, pixels which have value 1 may be considered to be the result of object motion or illumination changes. This assumes that the frames are properly registered.

A straightforward domain-independent method for comparing regions in images is to consider corresponding areas of the frames. These corresponding areas may be the super-pixels formed by pixels in non-overlapping rectangular areas comprising m rows and n columns. The values of m and n are selected to compensate for the aspect ratio of the camera. A frame maybe partitioned into disjoint super-pixels or use a local mask and compare the intensity distributions. One such method is based on comparing the frames using the likelihood ratio computing over such super-pixels or windows. Thus, we may compute

$$\Lambda = \frac{[(\sigma_1^2 + \sigma_2^2)/2 + (\mu_1 + \mu_2)/2]^2}{\sigma_1^2 + \sigma_2^2}$$

(where μ and σ^2 denote the mean gray value and the variance for the sample areas from the frames, and then use

$$DPjk(x, y) = 1 \text{ if } \Lambda > T$$
$$= 0 \text{ otherwise.}$$

.

The likelihood ratio test works quite well for most real world scenes. In Figure 13, we show two frames of a sequence and the difference pictures for those. This figure shows the first frame in which it is known that none of the moving objects are there and the second frame is a normal frame. As can be seen, by considering the first frame as the one containing only the static components or the background, a simple subtraction mechanism, commonly called background subtraction, gives the shape and location of all moving objects.

(a) A frame showing background in a scene.

(b) A later frame that has some moving objects.

(c) A difference of frame a and b shows the masks of moving objects.

Figure 13. Background subtraction results in detection of only moving objects.

Computing Image Flow

Image flow, commonly called optical flow, is the velocity field in the image plane due to the motion of the observer, the motion of objects in the scene, or apparent motion which is a change in the image intensity between frames that mimics object or observer motion. Image flow carries valuable information for analyzing dynamic scenes. Several approaches for dynamic-scene analysis have been proposed which assume that image flow information is available. Image flow is determined by the velocity vector of each pixel in an image. Several computational approaches have been devised for calculating image flow based on two or more frames of a sequence. These computational approaches can be classified into two general categories: feature-based and gradient-based. If a stationary camera is used, most of the points in an image frame will have zero velocity. This is assuming that a very small subset of the scene is in motion, which is usually true. Thus, most applications for image flow involve a moving camera.

Although image flow has received a significant amount of attention from researchers, the computing techniques developed for image flow do not produce detailed results of the quality which will allow the valuable information to be recovered. Image flow computations have found applications in compression techniques where one does not require the flow vectors at the same precision level as for recovering dynamic information.

Tracking

In many applications, an object must be tracked over a sequence of frames. If there is only one object in the sequence, the problem is easy to solve. In the presence of many objects moving independently in a scene, tracking requires the use of constraints based on the nature of objects and their motion. Due to inertia, the motion of a physical entity cannot change instantaneously. If a frame sequence is acquired at a rate such that no dramatic change takes place between two consecutive frames, then for most physical objects, no abrupt change in motion can be observed. The projection of a smooth three-dimensional trajectory is also smooth in the two-dimensional image plane. This allows us to make the smoothness assumption in images. This property is used to formulate *path coherence*. Path coherence implies that the motion of an object at any point in a frame sequence will not change abruptly. We can combine the solution of the correspondence problem for stereopsis and motion. The following three assumptions help in formulating an approach to solve the correspondence problem:

- The location of the given point will be relatively unchanged from one frame to the next frame.

- The scalar velocity of a given point will be relatively unchanged from one frame to the next.
- The direction of motion of a given point will be relatively unchanged from one frame to the next frame.

We can also use the smoothness of image motion in monocular image sequences. Based on such assumption, many computational approaches have been developed to track object or points on objects in videos. This is a very important research area and research has resulted in a rich set of techniques for different applications.

Further Readings

Nagel [174] proposed the use of the likelihood ratio for motion detection. Much of the work presented here on difference and accumulative difference pictures was done by Jain with other researchers [129, 130, 122, 125].

Tamura et al took the approach of devising texture features that correspond to human visual perception [1]. Statistical features of grey levels were one of the earliest methods used to classify textures. Haralick [7] suggested the use of grey level co-occurrence matrices (GLCM) to extract second order statistics from an image. GLCMs have been used very successfully for texture classification in evaluations [2]. Turner [9] first implemented this by using a bank of Gabor filters to analyze texture. A bank of filters at different scales and orientations allows multichannel filtering of an image to extract frequency and orientation information. This can then be used to decompose the image into texture features. Our implementation is based on that of Manjunath et al [10,11].

Index Terms

Exercises

References

Chapter 21: Content and Context

Human perceptual system has been a topic for active exploration by philosophers for long time. Almost two century ago [Ber] George Berkeley asked: "**If a tree falls in a forest and no one is around to hear it, does it make a sound?**" Sound is often defined as the sensation excited in the ear when the air or other medium is set in motion. Thus, if there is no 'receiving ear' than there is no sound. In other words, perception is not only data – it is a close interaction between the data, transmission medium, and the interpreter. This is shown in figure 1.

Multimedia communication and computing is fundamentally related to the perception problem. In any perception problem, there are three components that must be considered.

- The data acquired for an environment
- The medium used to transmit physical attribute to the perceiver
- The perceiver

Characteristics of each of these must be considered in designing and developing a multimedia system. It has been very well realized, and rigorously articulated and represented, that we understand the world based on the sensory data that that we receive using our sensors and the knowledge about the world that we have accumulated since our birth [Popper, Neisser]. Both the data and the knowledge are integral component of the understanding

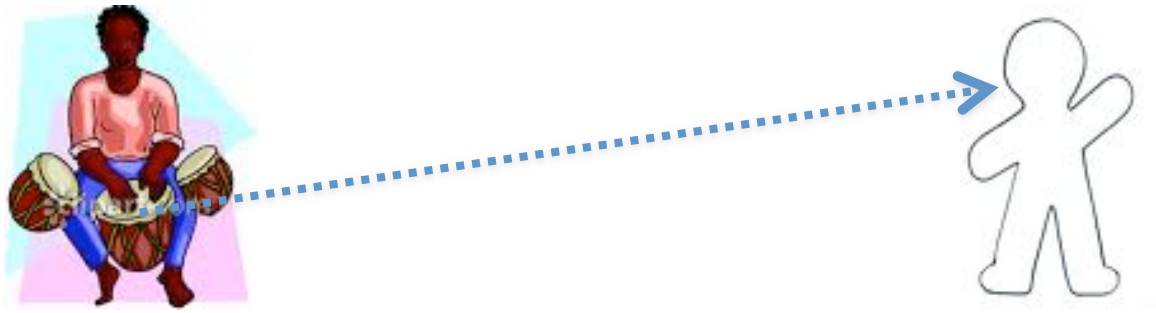


Figure 1: Perception requires a source, medium and a perceiver.

Let us revisit multimedia computing from fundamentals. Where does the multimedia data come from? Why do we even need multimedia? What is the multimedia content problem?

Multimedia data, such as visual (photos and video), aural, and other sensory data are captured for an event that unfolds over time. Each medium represents a particular physical spatio-temporal attribute of the event. An event represents changing relationships among objects and these are captured by different media. The data captured by any kind of sensor really represents these spa-

tio temporal physical attributes of the environment. Objects are part of the environment and their physical attributes are also captured by the media.

Each sensor only captures one type of physical attribute from its 'perspective' afforded from its physical location, including its orientation. Multiple sensors could be combined to create the composite data representing a synchronized signal obtained from these sensors. Thus, one uses appropriate number and types of sensors to capture all attributes of the event that may be of interest in a particular application. Multimedia is the right approach to capture event information and experiences because each medium captures only one physical attribute and taken as a whole, the multimedia stream is capable of combining the correlated and complimentary information from individual streams to provide more holistic information and experience than possible using any one medium. None of the individual medium, including the most powerful human senses (the vision), can capture holistic experience in most applications. Humans have five senses and combine them to experience events in real world.

Equally important is the fact that each sensor captures data about the environment from its position and perspective. If its position or perspective is changed, then the data and experience also change. For interpretation of the data, one must know the position and perspective. Moreover, many sensors, like cameras, have several other parameters (focal length, aperture diameter, flash, etc.) that determine the capture of the data and hence they are very important in understanding and analyzing the experience represented by the data.

Most important component in multimedia computing systems is the user. Each user is unique and while interacting with a system, the context may be different. Interpretation of the data is not only user dependent but also dependent on the context of the user. If you give the same photo to different people and ask them to assign tags to represent the photo, there may be as many different tags as the number of people assigning tags. Moreover, many studies have demonstrated that if you give the same photo to the same person at two different times in different contexts, then the tags assigned are different. The concept of Rorschach tests is based on the theory that an interpretation of data is as much, or more, dependent on the person than the data.

Connecting Data and Users

Multimedia computing addresses a problem that many other fields like computer vision, databases, and information retrieval face: connecting data and users. As shown in figure 2 below, data exists in many forms, ranging from bits to alphanumeric documents to photos and video. On the other hand users of the data in a modern computing environment may come from many different education backgrounds, of different culture, and of different socio-economic status.

The challenge is how to connect a user with a data source so the user can use the data he needs to solve his application. A key point to remember is that a user is never interested in what and where the data is; she is only interested in solving the problem at hand.

Figure 2: A user normally combines information from multiple sources to form a holistic concept about an object or event.

The major hurdle in connecting users to the data is often referred to as the semantic gap, as stated in [11]:

"We opine that most of the disappointments with early retrieval systems come from the lack of recognizing the existence of the semantic gap and its consequences for system set-up. *The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation*. A linguistic description is almost always contextual, whereas an image may live by itself."

To understand semantic gap, let's consider figure 3. This figure shows that the data operations in a computer start at the bit level and can be structured to represent various data concepts such as documents, images, and videos. A user on the other hand always starts thinking in terms of objects and events and builds other concepts based on the basic notion of objects and events. The transformation of data level concepts such as images to user level concepts such as objects and events is the challenge that must be solved by content analysis and used in content organization and retrieval.

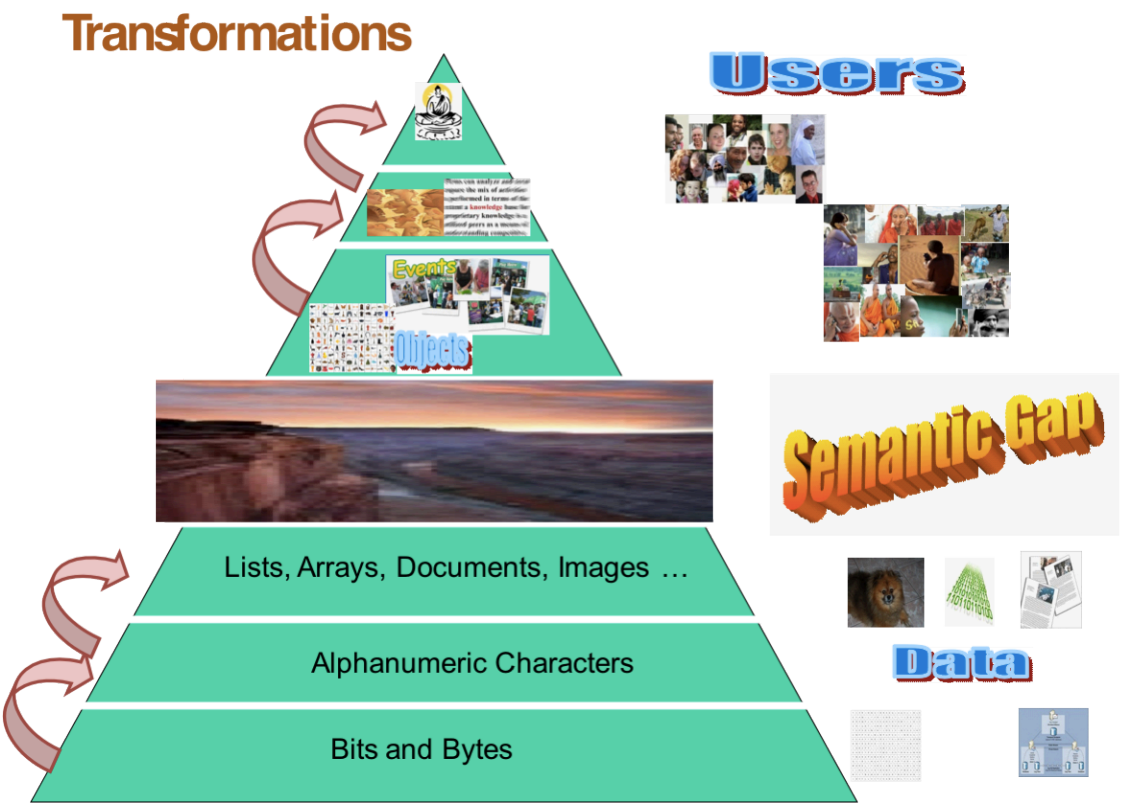


Figure 3: Semantic gap is created because computer operates using bits and structuring them into bytes and other structures. Humans think in terms of events and objects and build their concepts in terms of those. How do we go from bits to objects and events is the gap that must be bridged to allow smooth interactions between humans and computers.

Content and Context

Content usually refers to data values in a multimedia stream, such as images or audio. It is commonly understood that we refer to substantive information perceived by a user in the data that is represented by a particular file. Content analysis and content-based retrieval represents majority of research done in multimedia and related fields like computer vision. Thus a photo may contain a person standing near a car next to a house. The challenge faced by content analysis is the fa-

mous problem of pattern recognition. In all sensory data, the problem is to segment the data into meaningful parts and to use known models of objects of interest to label all segments of an image. This is where one runs into a tricky situation: We need segments to recognize objects but we also need objects to segment the data. It is possible to formulate this problem such that one can use models of potential objects to segment and then see how best the segments fit object recognition. One may potentially use an optimization framework to accomplish this. The problem gets complicated and almost intractable because in some cases, most notably in images, a higher dimensional space is mapped into 2-dimensional space resulting in loss of information making the problem impossible to solve unless some strong assumptions are made. This is the reason behind the sensory-gap [11] defined as: *The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene*.

In many sensors the signals from multiple objects get intermingles and combined making it almost impossible to solve the problem. The only way to simplify the problem appears to be to use other information and reduce the number of potential objects that could be in the data and and then check which objects are most likely to result in the signal.

Philosophers and scientists have been trying to solve the mystery of human perception for several centuries and are still far from any seemingly right solution. Closer to multimedia, people have been working on image recognition and speech recognition (notice only speech recognition, not audio recognition) and are still far from being close to solving these problems even with the powerful computing infra-structure that we have. The successful solutions usually are for limited domains meaning the number of objects is limited in those applications, making the problem more tractable.

Let us look at a related concept: context. Context is defined in standard dictionaries and reference sources as:

- The set of circumstances or facts that surround a particular event, situation, etc.
- The interrelated conditions in which something exists or occurs: environment or Setting.
- Determinant of meaning.

In technical areas, context started receiving attention in the last decade and has been receiving increasing attention. The precise definition of context is very difficult because content and context are closely related, as one could easily see from the dictionary meaning of context. The role of context in understanding our environment has been emphasized by several researchers in psychology of perception, neurophysiology, and cognitive sciences. Clearly, a perceiver is at least

as important as the data in perception. The perceiver uses extensive context in understanding signals and recognizing objects and events in it.

When using multimedia, each media data stream may act as context for other. Consider a video. It has an image sequences and an audio stream. As is easily obvious, without audio we can not fully understand video and without visual signal we can not fully experience video. In multimedia, each media stream has incomplete information about the environment captured and must utilize complementary information in some other media stream to recreate the environment to understand and enjoy it.

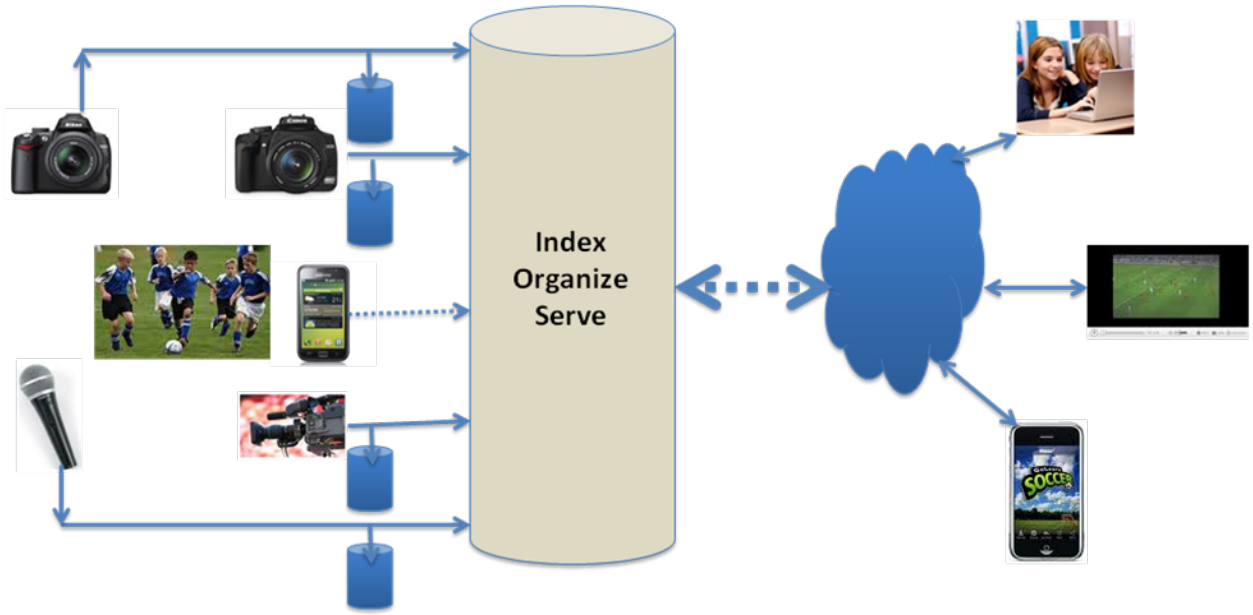


Figure 4: An eco-system showing creation to consumption of multimedia data. Capture depends on the location, environmental conditions and device parameters. All data is locally stored and collectively organized and indexed for serving consumers on different device.

Types of Context in Multimedia Systems

The context or knowledge that could be used in analyzing data may be considered in following different classes:

1. **Context in Content:** Relationship among different objects and even in their subparts in real world can be converted to relationships among data items and can be applied in analysis of data.
2. **Device Parameters:** Environmental parameters of the digital devices at the time of data

collection play important role in the analysis of data.

3. **Data Acquisition Context:** Knowledge about the person collecting data, location, and environmental conditions at the time of data acquisition (sun angle, cloudy, rainy, night, indoor, etc.) affects the content and knowledge of these could be used in data analysis.
4. **Perceiver:** Cognitive scientists know the importance of the perceiver. Rorschach tests are a clear demonstration of the knowledge and personality of the perceiver in interpretation of visual data.
   Interpretation Context: Real world situation in which the data is interpreted results in focus on different aspects of data. A botanist looks at the garden with different goal and interprets it differently than a person interested in enjoying the beauty of flowers.

Consider a simple case to understand how context can significantly simplify analysis: A photo is taken and needs to be interpreted. If one knows when the photo was taken and at that time what was the lighting level in the scene, one could use appropriate parameters for segmentation and interpretation of images. Moreover, if the photo was taken in Iowa, one should not expect beaches or mountains.

In the following, we discuss the role of each of these types of context. Explicit consideration of the above classes of contexts helps analysis and management of multimedia data. In the following we show some examples of these contexts.

Context in Content

In image understanding, one starts with the intensity values at every pixel and computes edges, regions, and other features. These features are then used to find meaningful objects and events in the image. The role of context in image understanding, commonly called computer vision, was emphasized even in very early days by systems developed for the block worlds and those for indoor scenes like MYSYS. Waltz [4] developed a constraint propogation framework to label junctions and understand configuration of blocks. This work was one of the earliest work in computer vision and the one that also resulted in starting to explore role and propagation of constraint.

A very clear role of context was demonstrated in 1976 in MYSYS system [8], where it is stated: "In scene analysis, it is frequently impossible to interpret parts of an image taken out of context. Different objects may have similar appearances, while objects belonging to the same functional class can have strikingly different appearances (e.g., chairs). Ambiguous local interpretations must be ruled out by using contextual constraints to achieve a meaningful, globally consistent interpretation of the whole scene."

Further they said: "Scene interpretation is an attempt to explain observed sensory data in terms of prior knowledge about the depicted domain. The explanation can entail many types and levels

of knowledge, some of which may be probabilistic or inconsistent. It must also allow for the likelihood that the data is noisy. For these reasons, scene interpretation is not a purely deductive problem with a unique correct solution; it is a problem that requires a search for the best or optimum explanation."

Early research in scene understanding resulted in realization that problems in scene understanding are posed by providing (1) a set of possible assignments for each region, with associated a priori likelihoods for each assignment and (2) a set of constraints that are derived for the types of scenes to be analyzed and that determine the a posteriori likelihood of regions.

The above approach, resulted in popular framework commonly called relaxation labeling, has been formalized and used in many computer vision systems. The basic idea in this approach is to utilize knowledge of local relationships among objects that may appear in a scene and use these local relationships to propagate local interpretations repeatedly to refine overall interpretations over the image. Thus one may use simple facts like 'a computer monitor is on a desk', 'floor is at the bottom in an image', and 'desk is on the floor' in an office scene. A set of such constraints among all objects can then be used iteratively in the interpretation process. The process starts with recognition of all possible regions and assigning them all plausible levels. The relaxation process then iteratively eliminates all implausible levels. When this process terminates, each region is assigned best possible interpretation based on the constraints, or the knowledge, available.

Relaxation processes have been defined at pixel levels. Also the relaxation processes have been developed considering a set of labels such that each iteration eliminates certain labels. Another approach uses probabilistic assignment of labels by updating the probability of labels in each iteration.

Verification Vision

Another commonly used approach for recognizing objects using context is called verification vision. Suppose that we are given an image of an object and we need to find how many times and where this object appears in an image. Such a problem is essentially a verification, rather than an object recognition, problem. There are many approaches for verification. One of the most popular one is called template matching. A verification algorithm can be used to exhaustively verify the presence of each model from a large model-base, but such an exhaustive approach will not be a very effective method. A verification approach is desirable if one, or at most a few, objects are possible candidates. If one is given some contextual information then that

could be used to constrain number of potential objects in an image. Thus, if one knows that an image was taken inside the kitchen of a western home, then one can list number of objects that may need to be verified; clearly an elephant or a tall banyan tree will not be in that list.

Context Only Image Search: Commercial Systems

The most commonly used example of use of context in search are the commercial application of image search from any major search engine, Google, Bing, or Yahoo. Suppose that you search for images with keyword Obama, rose, Tendulkar, or cars. Look at the results of the image search and you will be surprised. Most of the results are correct. Surprisingly, as is well known, most of these results are obtained without even opening the image file – that is without even looking at the content. These search systems only use the context provided to them from sources such as the name of the file containing picture, surrounding text on the page where the picture file appears, and the topic of the page. These search engines perform much better than any content based retrieval system that we have seen, including the ones that one of the authors was involved in developing.

Device Parameters

For considering a common example of device parameters, lets consider a device that most of us use regularly: a digital camera. Unlike their predecessors, modern digital cameras are more 'event capture' devices rather than photographic devices. Modern digital cameras have multiple sensors, including GPS to determine the location of the photograph, present on it that can capture much more information about the photo shooting event. The digital photograph is no longer just a collection of pixels; it has a host of sensory information stored as metadata. This information can be summarized if we model the digital photograph as a multilayered structure as shown in Figure 5. The layers are: i. Pixel/ Spectral layer and ii. Meta Layer. The pixel layer stores information recorded by the CCD (as pixel values). The content of the image is present in this layer. The Meta Layer stores the contextual information about a photo shoot. The Meta Layer can further be divided into the following sublayers: a. *Optical Meta Layer*. It contains the metadata related to the optics of the camera; e.g., the focal length,aperture, exposure time etc. These metadata store important cues about the context in which the image was shot (like the lighting condition, depth of field and distance of subjects in the image). b. *Temporal Meta Layer*. It contains the time stamp of the instant in which the photo was shot. The time stamp of a single image in a standalone environment is not informative enough.

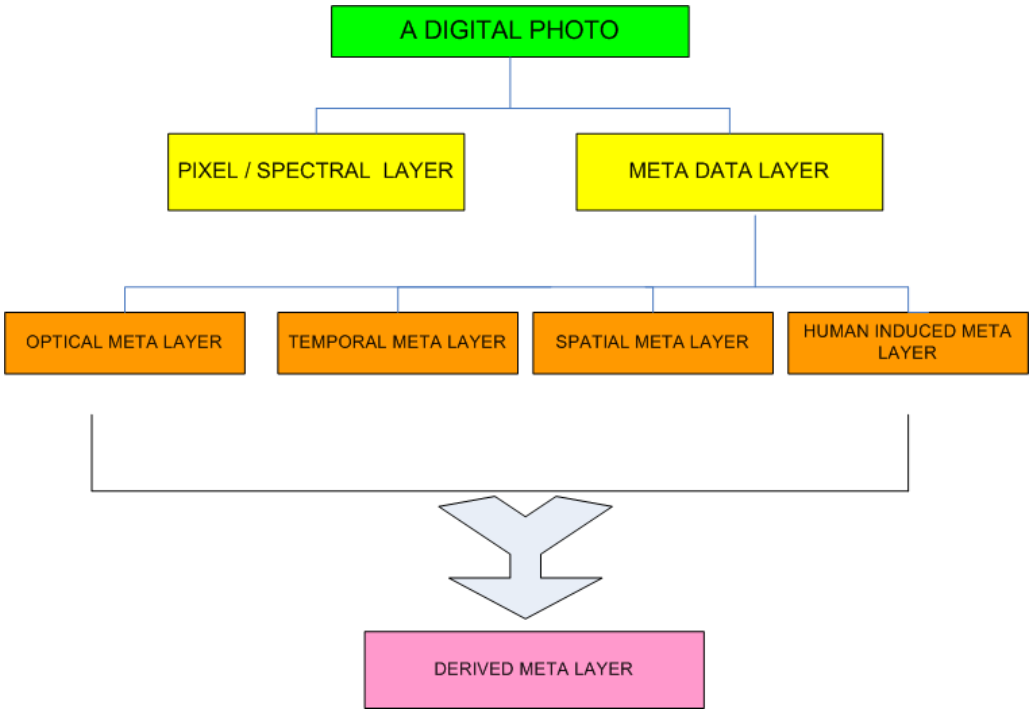


Figure 5: Different types of information stored with an image taken by a digital camera.

But in a collection of images (e.g., photo albums) the time difference can shed valuable light on the content of the images. c. *Spatial Meta Layer*. It contains the spatial coordinates of the places where pictures were shot. These coordinates are generated by the GPS systems attached to the camera. In case of camera phones, cell-tower ids help generate this data. d. *Human Induced Meta Layer*. This layer contains the tags/ comments/ ratings posted by people. Community tagging (in online photo albums) or voice tagging in mobile phones help generate data for this layer. e. *DerivedMeta Layer*. This metadata is inferred from other information by various statistical modeling approaches. The taxonomy defined here helps us to explore the information sources present in a digital camera image. Presently, the spectral, optical and temporal layers are present in almost all digital photographs, while the spatial, human induced and derived meta layers might not be present.

The Exchangeable Image File (EXIF) Standard specifies the camera parameters recorded during a photo shoot. The actual parameters recorded depend on the particular camera manufacturer. But there are certain fundamental parameters which are recorded by all popular camera models. These are: Exposure Time, Focal Length, F-number, Flash,Metering Mode and ISO. Subject distance is an important parameter which can be used to make important inferences about the image content. However most camera models do not store this parameter.

EXIF data is attached to all digital pictures and contains very valuable information about camera parameters used in taking photos. These parameters affect the part of the scene imaged and the intensity values of pixels, while others give very valuable contextual information about the data acquisition context. For a set of 2000 images, clusters of the photos based on the Optical Metadata associated with them provide valuable insights in the role of these parameters. The following figures show the probability distribution of human induced tags in different Exif based clusters.

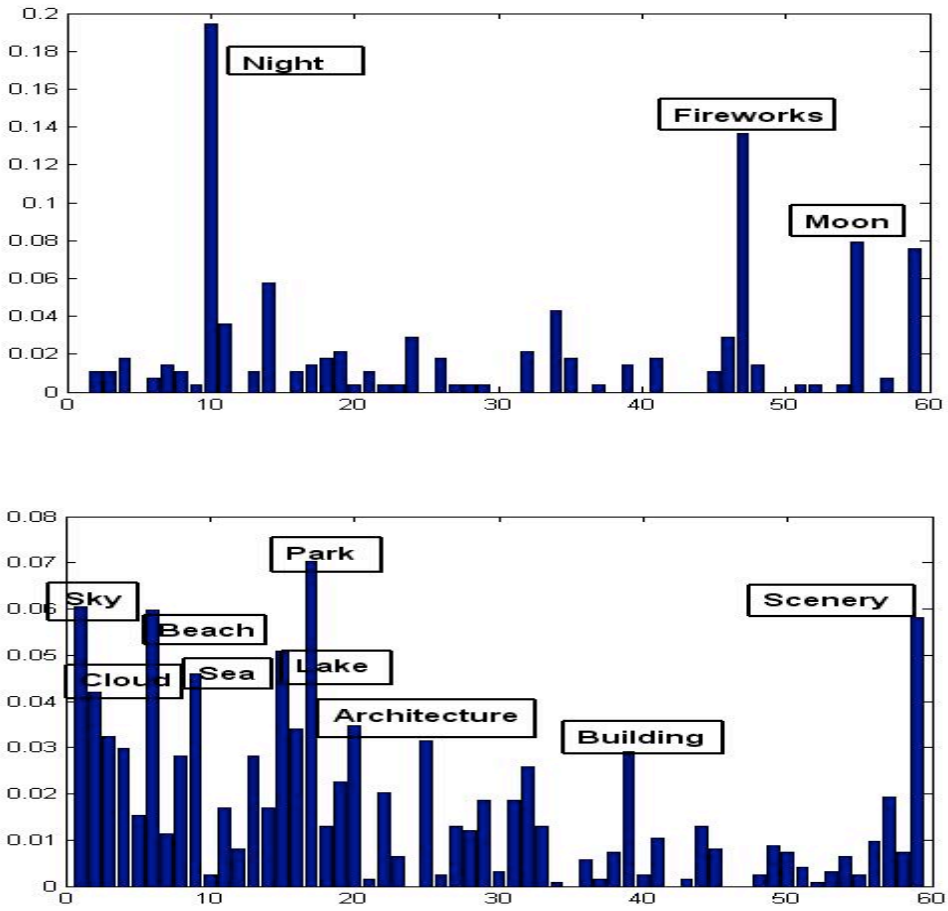


Figure 6: a cluster with No Flash and Large Exposure time

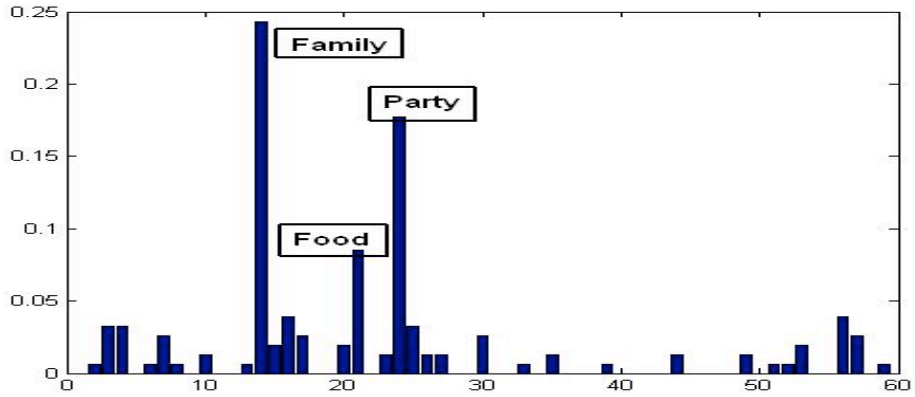


Figure 7: A cluster with NO Flash, Low Exposure and Low Focal Length (large Field of View)

Figure 8: a cluster with Flash, Larger Exposure (mostly indoor concepts)

Sometimes the context provided by device parameters like EXIF are more meaningful than those provided by the analysis of content or intensity values. Consider the photo shown in Figure 9.

If we try to assign tags based on pixel features only, we may obtain: *Scenery, City Streets, Illuminations, Wildlife*.

However, tags based on Exif features give: *Indoor, Party, Portrait, Group Photo Indoors*.

Why is there a discrepancy between tags predicted by the content and the context channel? This is a Photo of a Photo. The image originally appeared in a magazine and Figure 9 is a photo of it using a standard digital camera. Since the content and context channels capture two entirely different semantics about an image, the tags are so different.

Figure 9: What do you see in this photo. Exif parameters correctly analyze that this is a photo of a photo – this is not a photo taken in the real world.

Figure 10: Where is this picture taken. Photo interpretation will mislead that this is in China, but context correctly tells that this is taken in Orlando, Florida. GPS parameters have no difficulty in putting the photo at a right place.

Perceiver

A perceiver brings general knowledge of the environment that she lives to bear on the interpretation of the data. Since we all grew up in the physical world that we inhabit, we make several contextual assumptions about the world. Our brain automatically uses these assumptions in interpretation of data. The best example of this is in visual illusions. In visual illusions, we see something that is different from objective reality. Even after measurements and knowing that what we see is not right, our cognitive system does not accept the reality. This clearly demonstrates the role of assumptions that we all use in interpretation of data. Sometimes, the interpretation of data tells a lot more about the interpreter than about the data.

Rorschach tests

Many psychological tests and even many photos that commonly appear in psychology literature show a picture to people and them to specify what they see in the picture. In these experiments, the goal is to know about the personal characteristics of the perceiver. These tests demonstrate that the interpretation of data depends on the perceiver [16]. Common phrases used by people such as 'Do you see what I see in this picture?' are clear indication of the well recognized role of the perceiver.

Domain knowledge

Look at Figure X. Try to guess what this picture is. Now suppose you are tols that this is an Atomic Force Microsope image – now you will think about either cellular images or atomic level images of materials. In fact what you are seeing is a recent breakthrough in developing chemical sensors that could be developed cost-effectively [20]. Most possibly, you and I can not understand what this is because we do not know how to interpret these images because we don't have the context and associated knowledge. But this was an important image in the announcement of this breakthrough. What this shows is that without domain knowledge, it is almost impossible for people to analyze and understand content.

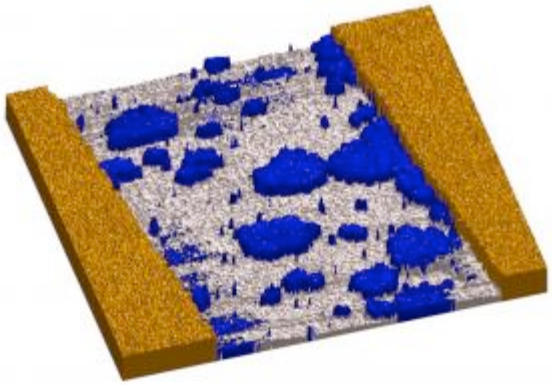


Figure 11: What do we see in this picture. It purely depends on the domain knowledge.

Context in Photo Management in SmartPhones

Next generation digital photos will be captured by smart phones. Sets. As shown in Figure 12, smartphones are privy to a lot of information sources which will help augment photos with lot of contextual knowledge. Let's consider some scenarios where context can play an useful role for organizing photos on smartphones.

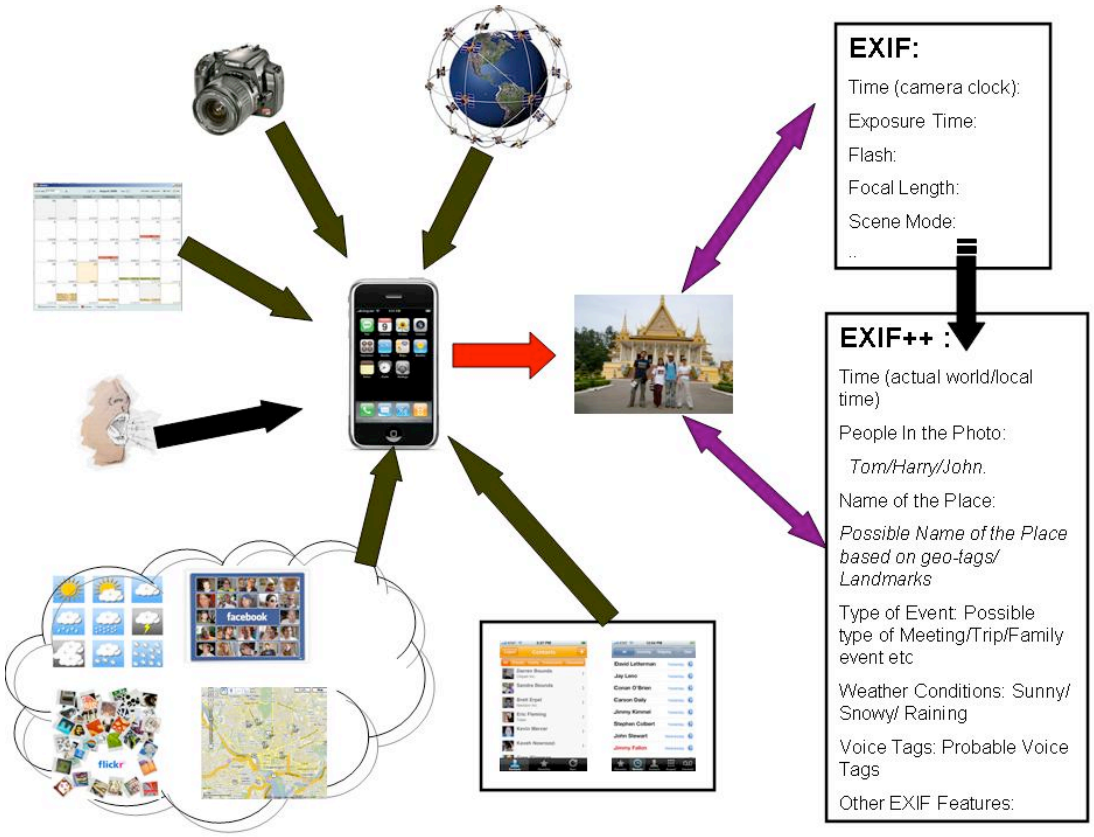


Figure 12: Showing how each photo by modern cameras will be augmented with very rich metadata – that we call EXIF++.

a. Identifying People: Most consumer cameras can detect frontal faces while shooting photos. However face recognition/assigning name tags to them is an open problem. In case of smart phones this problem can be reduced to a much easier face identification in a small set. Our personal calendars provide us with the name of people we are meeting at any particular point of time. If a portrait/group photo was shot at this time, we just need to compare with the faces appearing with the faces of people we were supposed to meet. The latter can come from any social

network or photo sharing site. If we do not good match, the system can go through the list of recent callers /callee and try matching with their faces. If not we can go through our contact list to find a good matching face. This makes the problem much easier to solve.

b. Identifying objects: Object identification is another problem. Usually people shoot landmarks or special objects of interest using their smart phones. The EXIF information can tell us if the object of interest is indoors or outdoors. We can also estimate a possible size of the object based on the focal length , field of view and subject distance. Geo location will help us narrow down to a small set of important objects (e.g., landmarks or flowers or food) which are shot in that area. Comparison to this much refined set is likely to generate much better results for object identification.

c. Event name tagging based on public /private Calendars: People shoot a lot of photos in their life events, e.g., parties, trips, meeting et al. It is very relevant and useful to tag photos based on the events. It may be very difficult to automatically tag a photo with an event name (John's birthday) or even a generic class name (indoor party) based on pixels and Exif alone. However, personal calendars can help on such cases to properly tag with event names. Further, people often participate in public events like concerts, baseball games, parades etc. There are abundant sources of event repositories on the web. Based on a users location information and the events taking place in the vicinity, it may be possible to predict the proper event name (e.g., Celtics vs Lakers Game) with good accuracy.

Further Reading

In human perception, Irwin Rock [5] and Gregory[6] have strongly championed the role of knowledge in many different forms in visual perception. They believed that context plays at least as important, in most cases significantly more, role as content. Human sensory processing uses context extensively. The role of context in understanding our environment has been emphasized by several researchers in psychology of perception, neurophysiology, and cognitive sciences [4,5,6,7,8]. A review of context as used in different computing systems and environments is provided in [3, 15, 16]. All digital cameras follow a meta-data standard called EXIF to capture the context around photos taken by the camera. A detailed analysis of these parameters is provided in [19].

Many philosophers and cognitive scientists, including one of the most noted in the 20th century Karl Popper [20] and Ulric Neisser [21] have created models of all human actions that include context and prior knowledge about an application as an integral component of understanding

data. Media processing techniques so far, however, have focused on content assuming that interpretation can be done based only on the data values. In semantic gap, an important point is the 'the linguistic description is always contextual'.

Context in general, and GPS in particular, has serious privacy implications. This is discussed in [18].

Index Terms

References

1. Sinha, P., Jain, R.: Classification and annotation of digital photos using optical context data. In: Proceedings of the 2008 international conference on Content-based image and video retrieval, ACM (2008) 309–318
2. Sinha, P., Jain, R.: Semantics In Digital Photos: A Contenxtual Analysis. In: Proceedings of the 2008 IEEE International Conference on Semantic Computing, IEEE Computer Society (2008) 58–65
3. Ye Sun Joung, Magda El Zarki, Ramesh Jain, "A User Model for Personalization Services", 3rd INTERNATIONAL WORKSHOP ON CONTEXT MODELING AND MANAGEMENT FOR SMART ENVIRONMENTS CMMSE, November 1-4, 2009.
4. Patrick Winston: The Psychology of Computer Vision (McGraw-Hill Computer Science Series), 1975.
5. Irwin Rock: The Logic of Perception (MIT Press) 1985.
6. Richard Gregory: The Intelligent Eye (1970). London: Weidenfeld and Nicolson.
7. Ramachandran, V.S. and Anstis, S.M.: The Perception of Apparent Motion, Scientific American, June 1986.
8. H. G. Barrow and Tenenbaum, J. M.: MSYS: A System for Reasoning About Scenes, Technical Report 121, AI Center SRI, April 1976.
9. Galleguillos, C and Belongie, S.: Context Based Object Categorization: A Critical Survey.Bo Gong and Ramesh Jain, "Segmenting Photo Streams in Events Based on Optical Metadata", Proc. IEEE Conf. on Semantic Computing, Irvine, CA, Sept. 17-19, 2007.
10. Santini, S. and Jain, R. "Beyond Query by Example; IEEE Second Workshop on Multimedia Signal Processing, Redondo Beach, CA , pp. 3 – 8, USA 7-9 Dec 1998.
11. Arnold Smeulders , et. al.,"Content-Based Image Retrieval at the End of the Early Years" IEEE Transactions on Pattern Analysis and Machine Intelligence_December 2000.
12. Exner, John E. *The Rorschach: A Comprehensive System*. Vol 1: Basic Foundations. New York: John Wiley & Sons. ISBN 0-471-55902-4, 1995.
13. Freuder, E.C. Recognition of Real Objects, Vision Flash, No. 33, MIT, 1972.

14. George Berkeley, *A Treatise Concerning the Principles of Human Knowledge*, 1734. section 45
15. G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In Proc. 1st Int. Symp. on Handheld and Ubiquitous Computing (HUC'99), pages 304–307. Springer-Verlag, 1999.
16. Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, Letizia Tanca, "A Data-oriented Survey of Context Models", SIGMOD Record, December 2007 (Vol. 36, No. 4)
17. J Kittler, J Illingworth, "Relaxation labelling algorithms--a review", Image and Vision Computing, 1985
18. G. Friedland and R. Sommer, "Cybercasing the Joint: On the Privacy Implications of Geo-Tagging", in Fifth USENIX Workshop on Hot Topics in Security (HotSec '10) at the 19th USENIX Security Symposium, Washington, D.C.
19. EXIF: Exchangeable image file format for digital cameras: Exif version 2.2. Technical report, Japan Electronics and Information Technology Industries Association, 2002.
20. Karl Popper, *Objective Knowledge: An Evolutionary Approach*, 1972, Rev. ed., 1979.
21. Ulric Neisser, *Cognition and Reality*: Principles and Implications of Cognitive Psychology Neisser, U. (1976) San Francisco: W. H. Freeman

Exercises

Why is it that at many places where they show video, such as in airplanes, they charge for renting headphones? You can see visual component of the video free but must pay for the audio component? Does this mean that audio has all information and video has none?

What is cocktail party effect? How is it that in such a noisy environment, people can carry conversations?

Chapter 22: Future Topics